Recurrent Neural Networks (RNN)

Deep Learning (DSE316/616)

Vinod K Kurmi
Assistant Professor, DSE

Indian Institute of Science Education and Research Bhopal

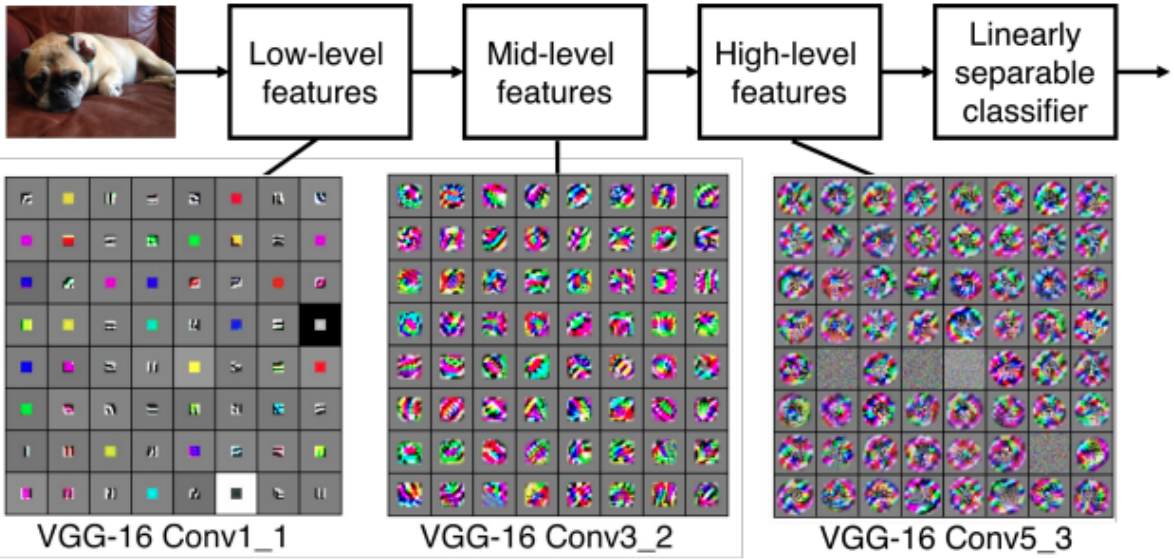
Sep 26, 2022

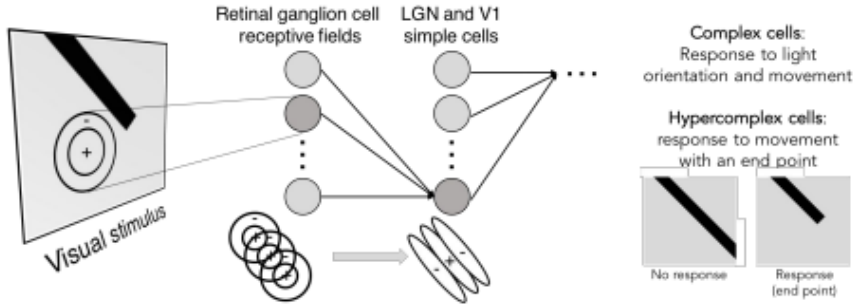
Disclaimer

- Much of the material and slides for this lecture were borrowed from
  - Bernhard Schölkopf's MLSS 2017 lecture,
  - Tommi Jaakkola's 6.867 class,
  - CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University
  - Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class
  - Hongsheng Li's ELEG5491 class
  - Tsz-Chiu Au slides
  - Mitesh Khapra Class notes

Previous class:CNN

Preview



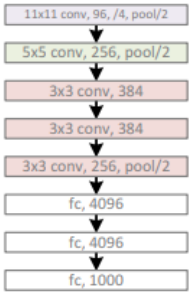


Previous class: CV models

-

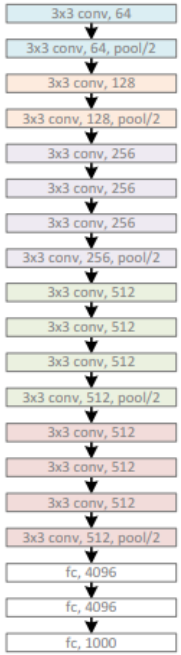
AlexNet, 8 layers

(ILSVRC 2012)



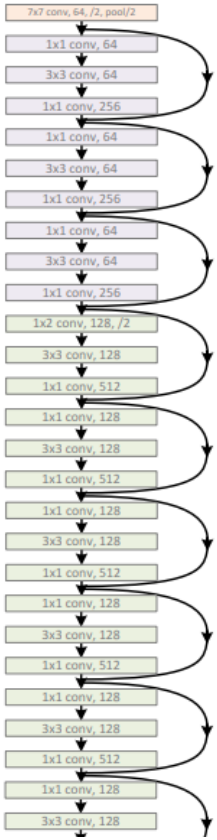
VGG, 19 layers

(ILSVRC 2014)

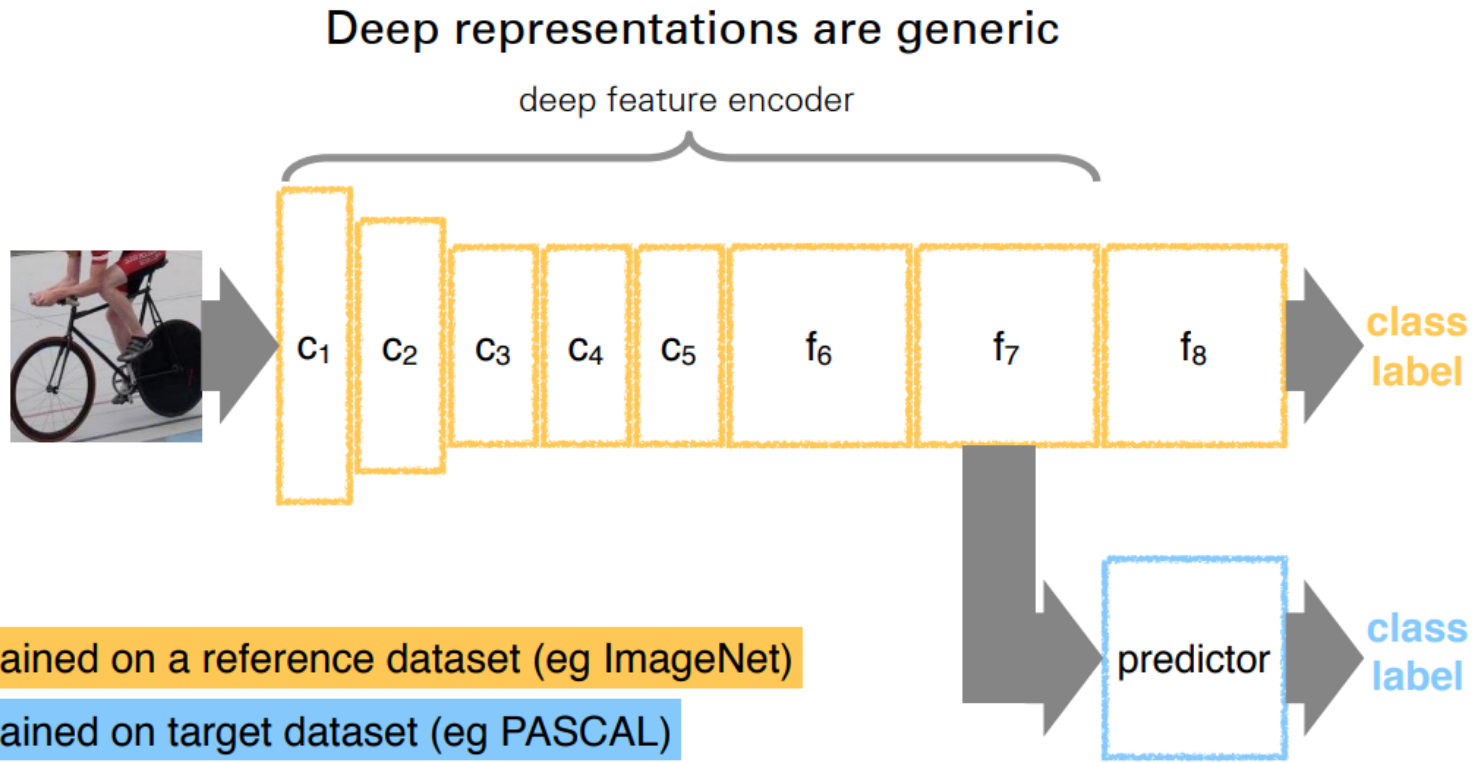


ResNet,
152 layers

(ILSVRC 2015)

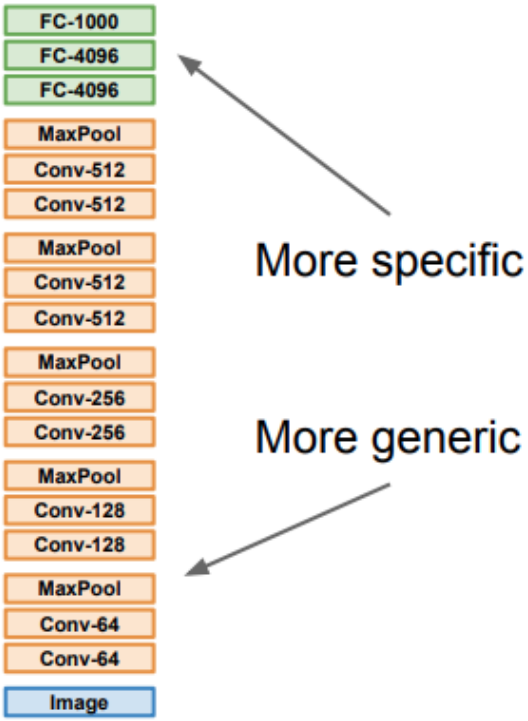


Previous class:



- A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

Previous class:Transfer Learning with CNNs

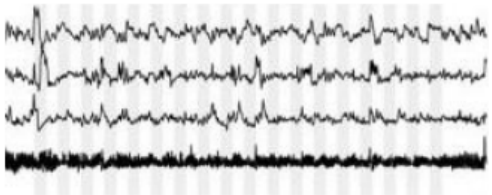


	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

How to deal the temporal things in DL

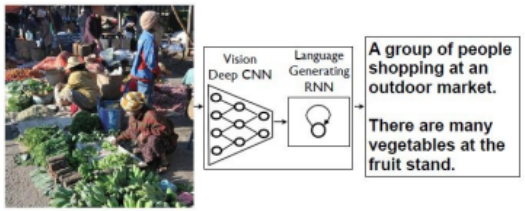
Sequence modeling

Sequential data

- "I took the dog for a walk this morning." — sentence
-  — medical signals
-  — speech waveform
-  — video frames

Modeling sequential data

- Sample data sequences from a certain distribution $P(x_1, \ldots, x_N)$
- Generate natural sentences to describe an image $P(y_1, \ldots, y_M | I)$



- Activity recognition from a video sequence $P(y | x_1, \ldots, x_N)$

Adapted from Xiaogang Wang

Modeling sequential data

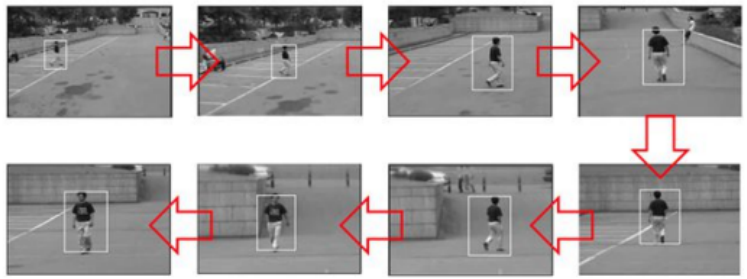
- Speech recognition

$$P(y_1, \ldots, y_N | x_1, \ldots, x_N)$$

→ Hey Siri

- Object tracking

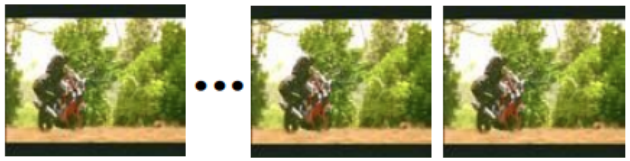
$$P(y_1, \ldots, y_N | x_1, \ldots, x_N)$$



Adapted from Xiaogang Wang

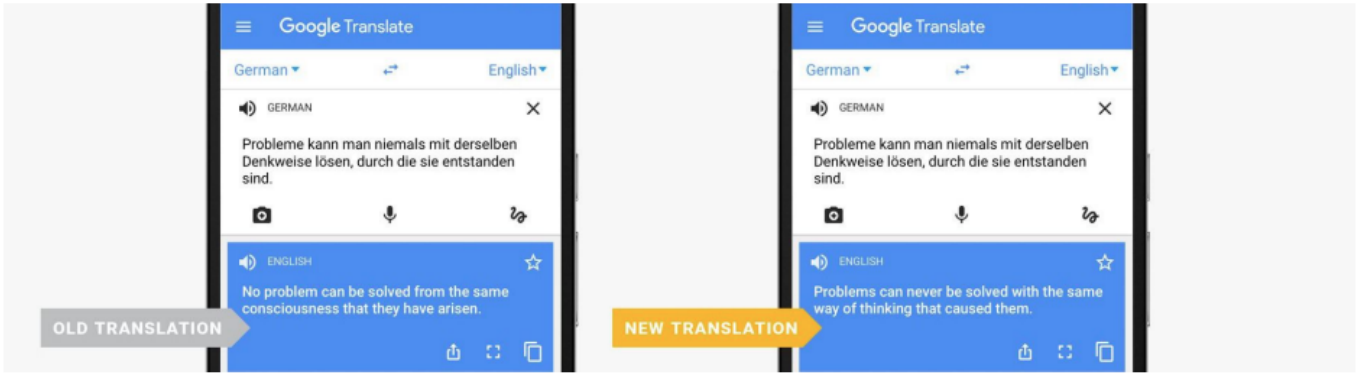
Modeling sequential data

- Generate natural sentences to describe a video $P(y_1, \ldots, y_M | x_1, \ldots, x_N)$



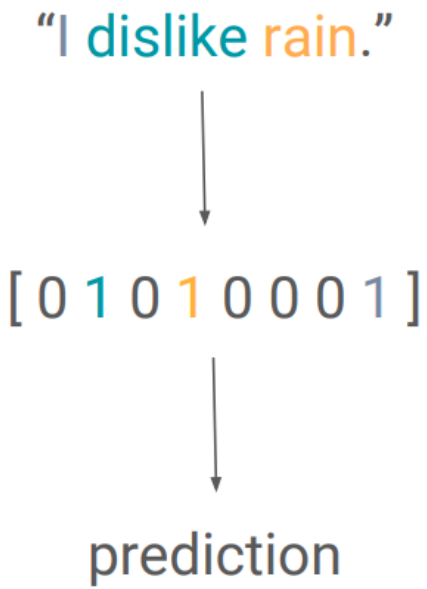
→ **A man is riding a bike**

- Machine translation $P(y_1, \ldots, y_M | x_1, \ldots, x_N)$

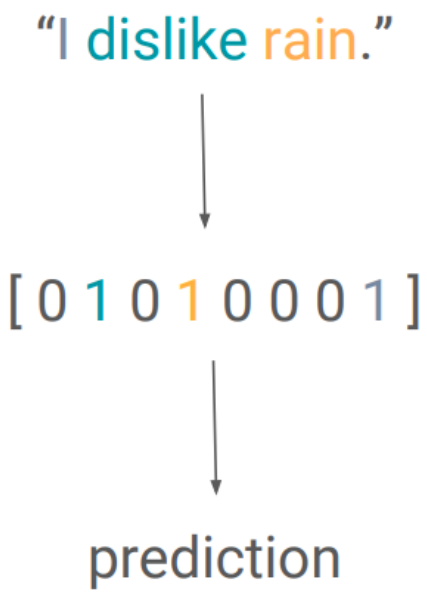


Adapted from Xiaogang Wang

Represent a sequence as a bag of words



Represent a sequence as a bag of words



- **Problem:** Bag of words does not preserve order

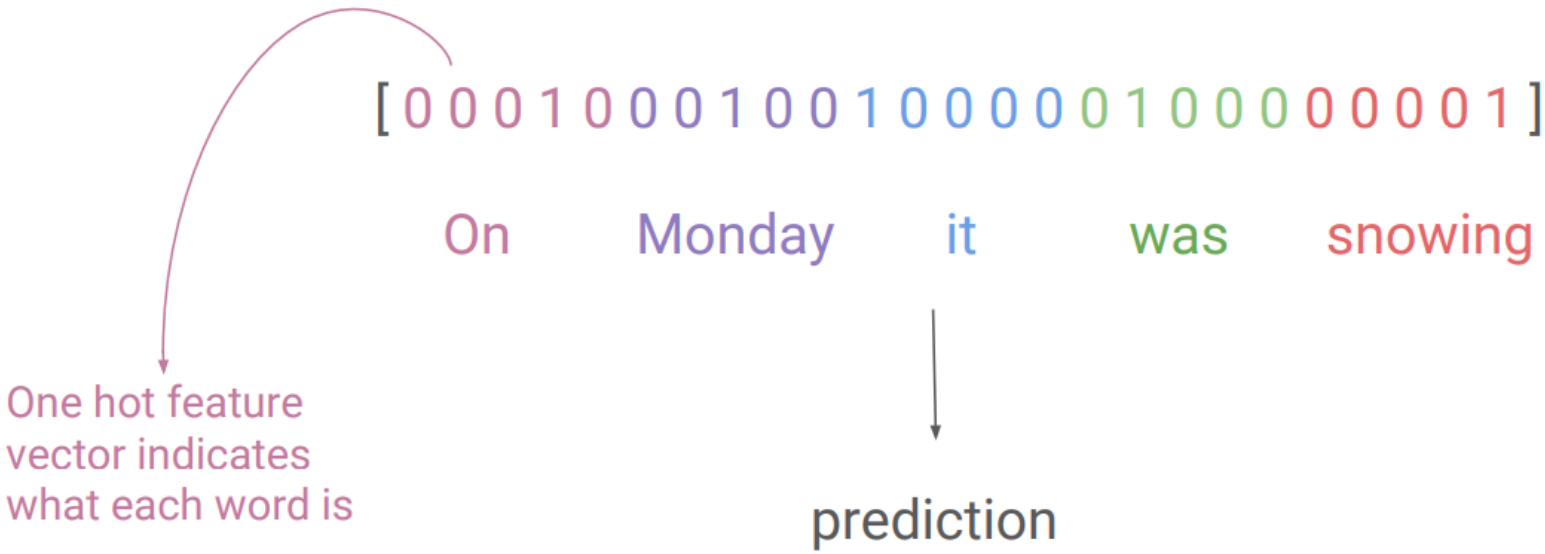
Bag of words does not preserve order!

"The food was good, not bad at all."

vs

"The food was bad, not good at all."

Maintain an ordering within feature vector



- **Problem:** Hard to deal with different word orders!

Hard to deal with different word orders!

"On Monday, it was snowing."

vs

"It was snowing on Monday."

Hard to deal with different word orders!

[0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1]

On Monday it was snowing

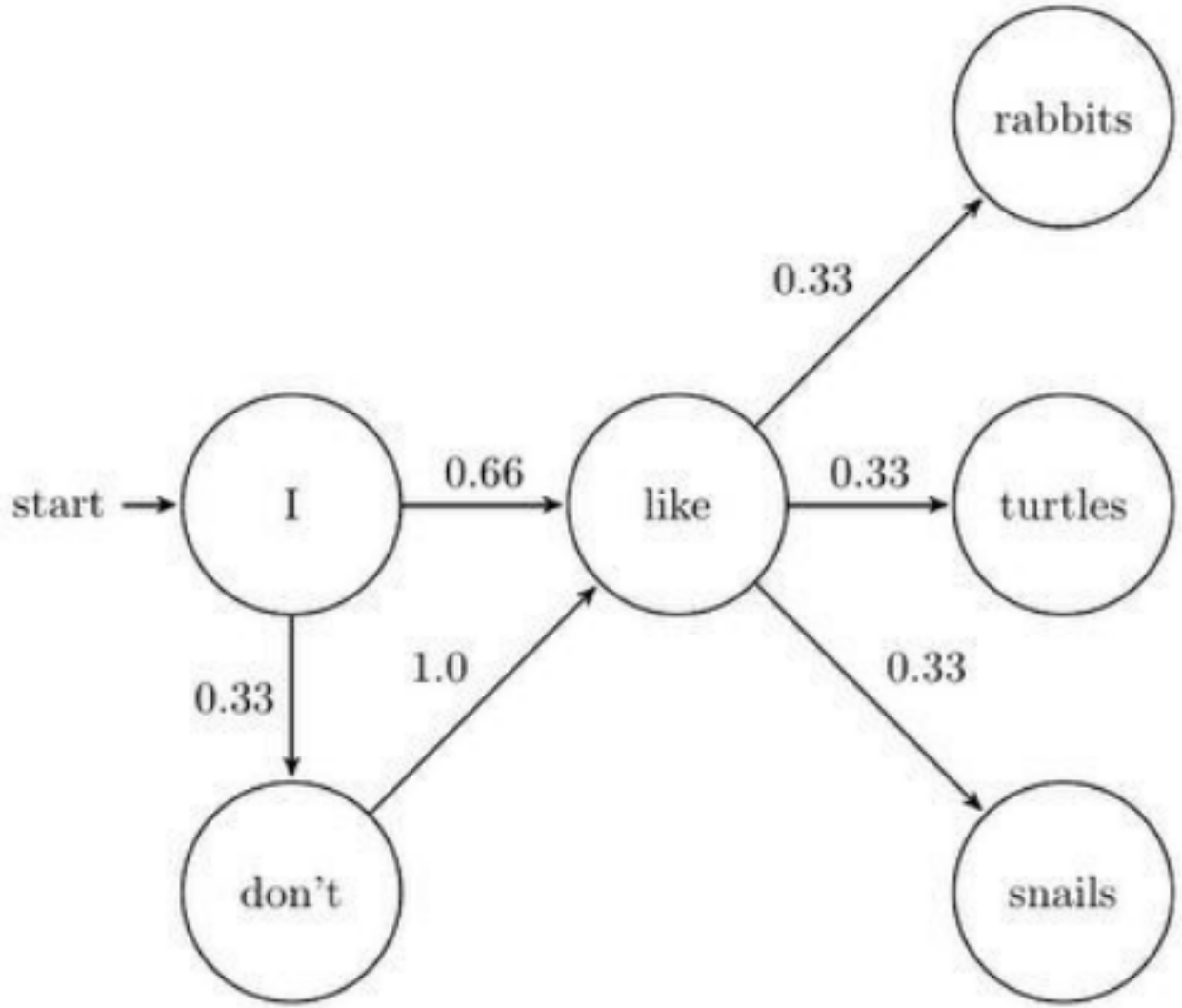
vs

[1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0]

It was snowing on Monday

- we would have to relearn the rules of language at each point in the sentence

Markov Models



Markov Models

- **Markov assumption:** Each state depends only on the last state.

"In France, I had a great time and I learnt some of the _____ language."

- We need information from the far past and future to accurately guess the correct word.

To model sequences, we need

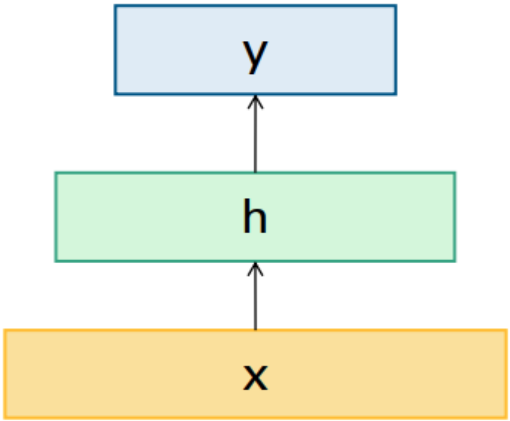
1. to deal with variable length sequences
2. to maintain sequence order
3. to keep track of long-term dependencies
4. to share parameters across the sequence

How to deal the temporal things in DL

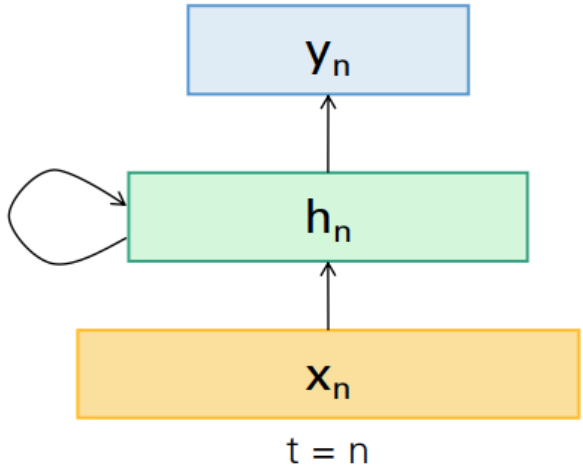
Recurrent Neural Networks

Recurrent Neural Networks

Feed Forward Network

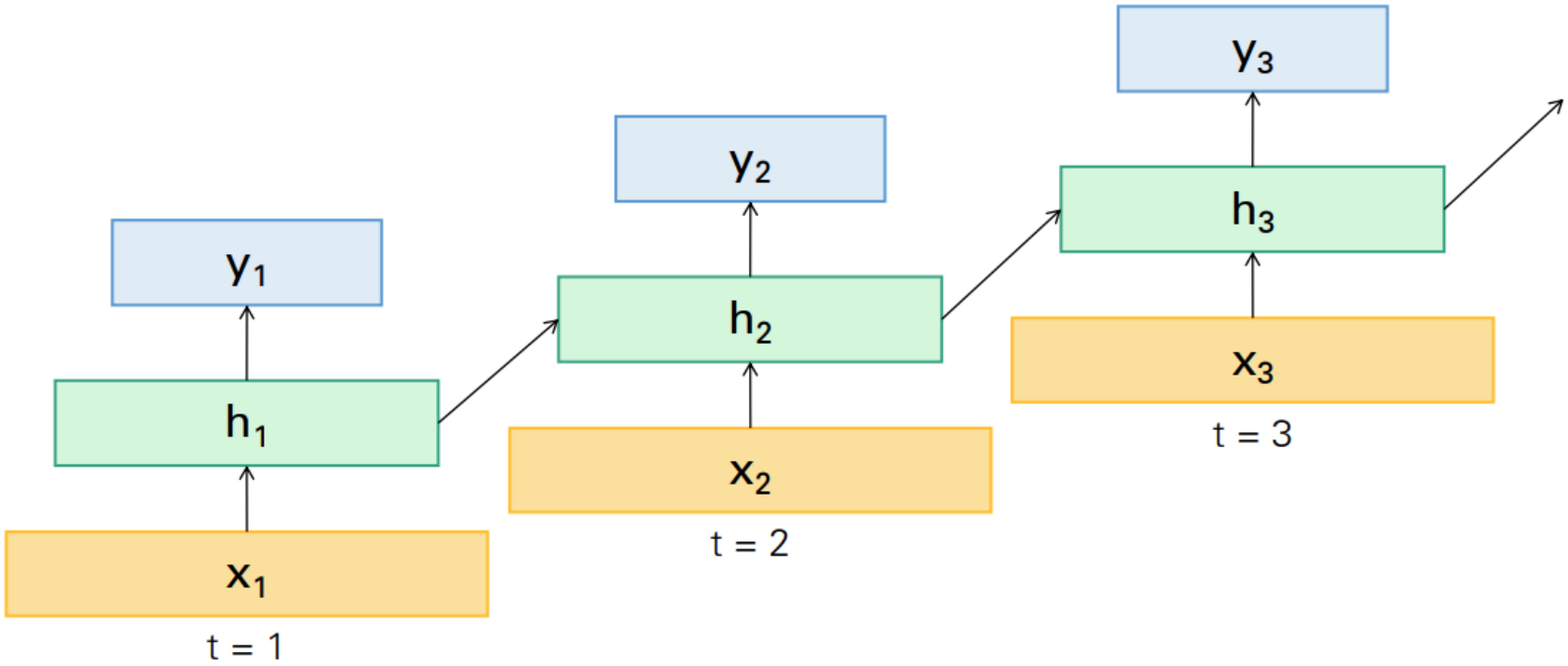


Recurrent Network

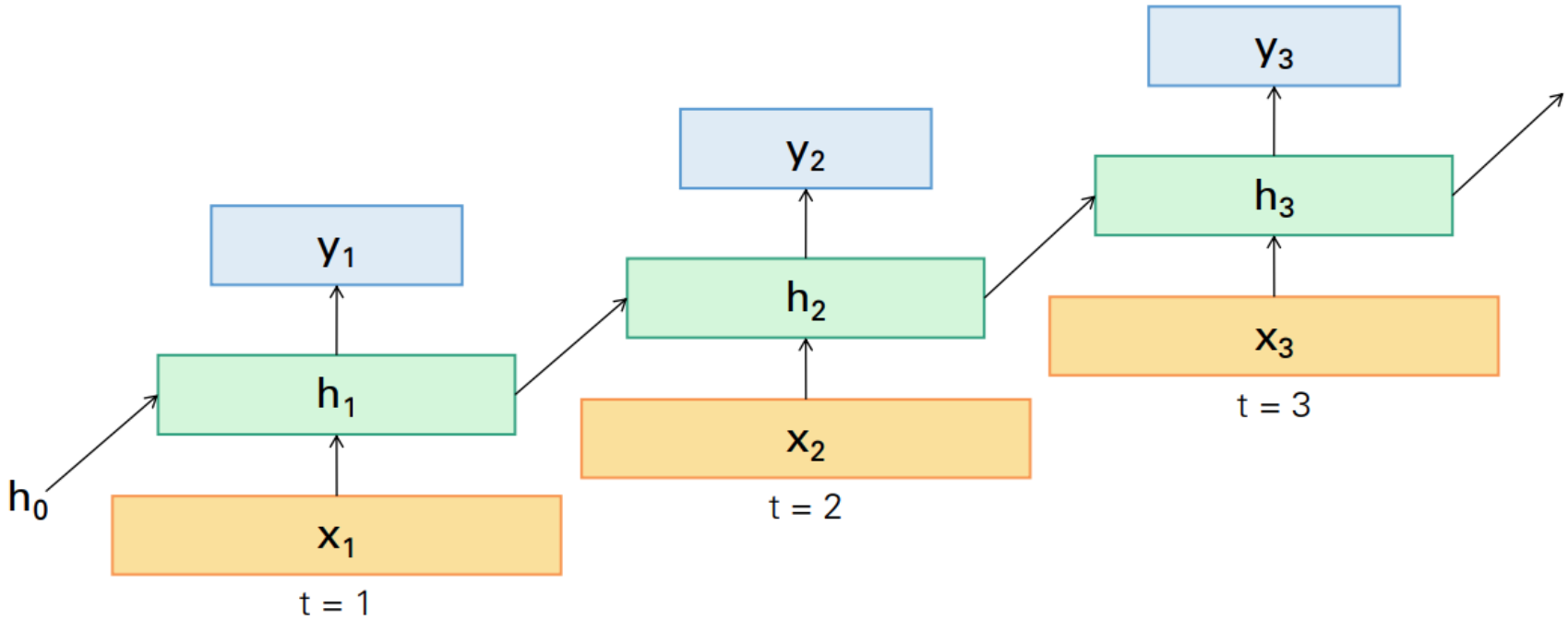


Notice: the same function and the same set of parameters are used at every time step.

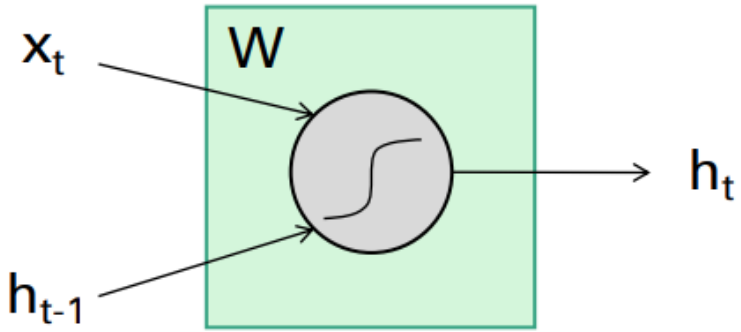
Unrolled RNN



Sample RNN



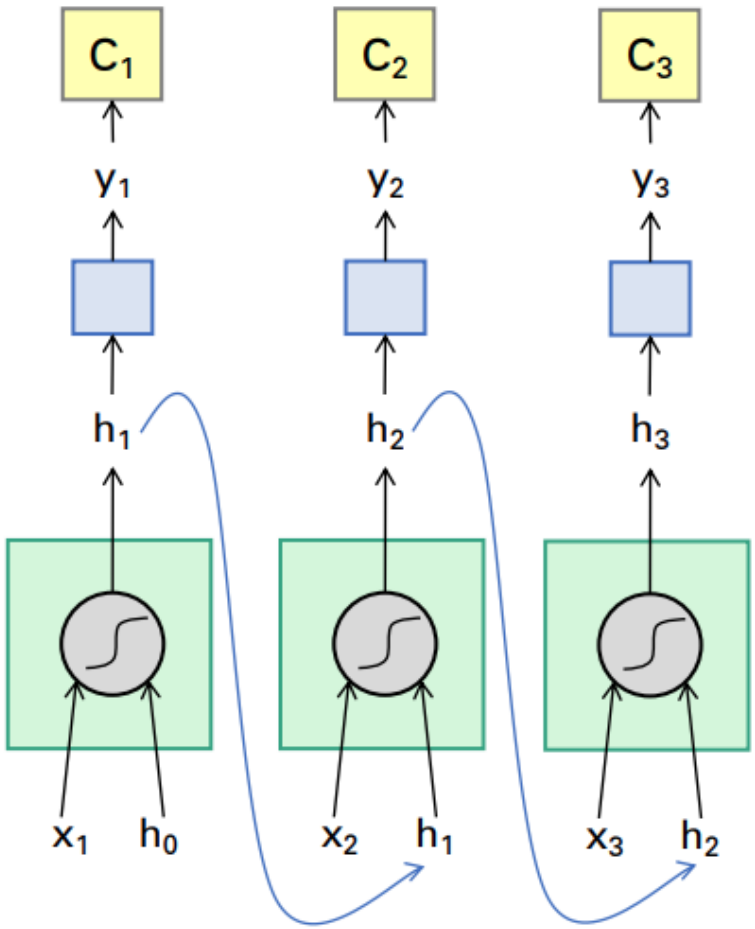
The Vanilla RNN Cell



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

cell state

The Vanilla RNN Forward

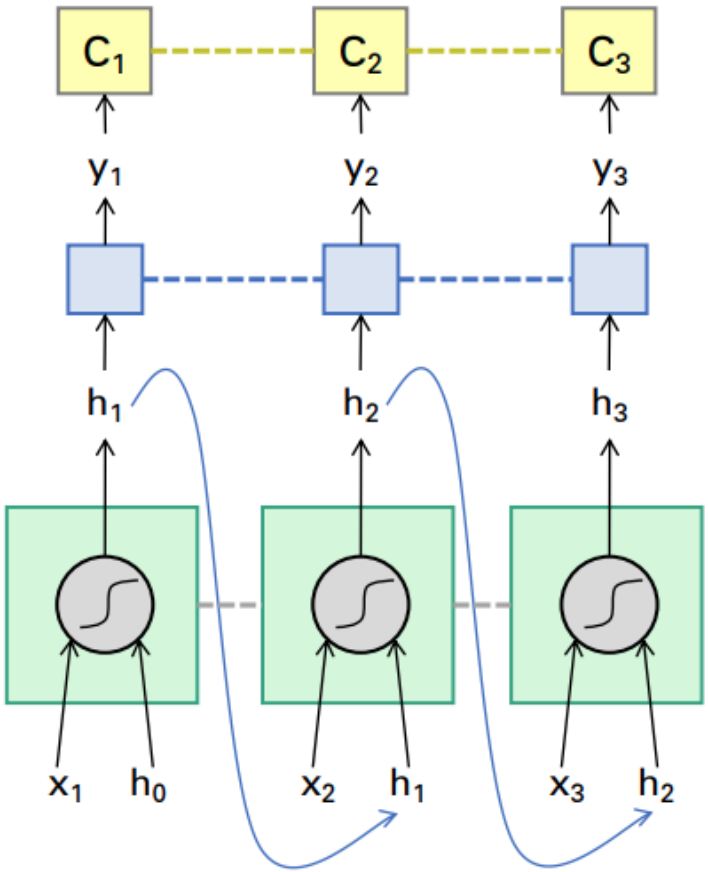


$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

The Vanilla RNN Forward



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

- Note that the weights are shared over time
- Essentially, copies of the RNN cell are made over time (unrolling/unfolding), with different inputs at different time steps

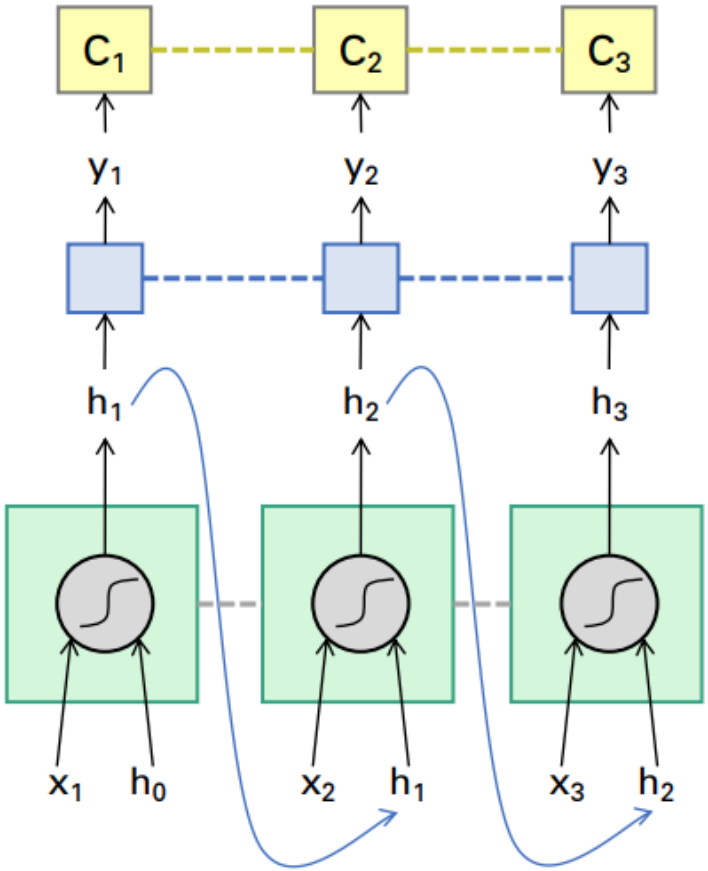
Applications or Examples

Sentiment Classification

Sentiment Classification

- Classify a restaurant review from Yelp! OR
  movie review from IMDB OR
  ...
  as positive or negative
- **Inputs:** Multiple words, one or more sentences
- **Outputs:** Positive / Negative classification
- "The food was really good"
- "The chicken crossed the road because it was uncooked"

Sentiment Classification



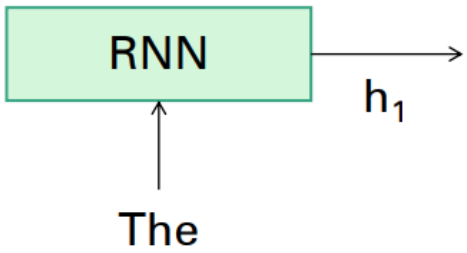
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

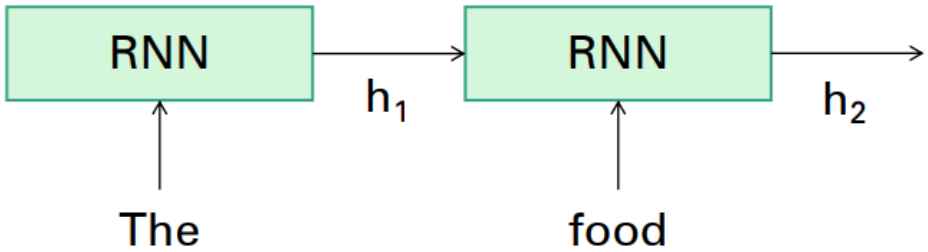
$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

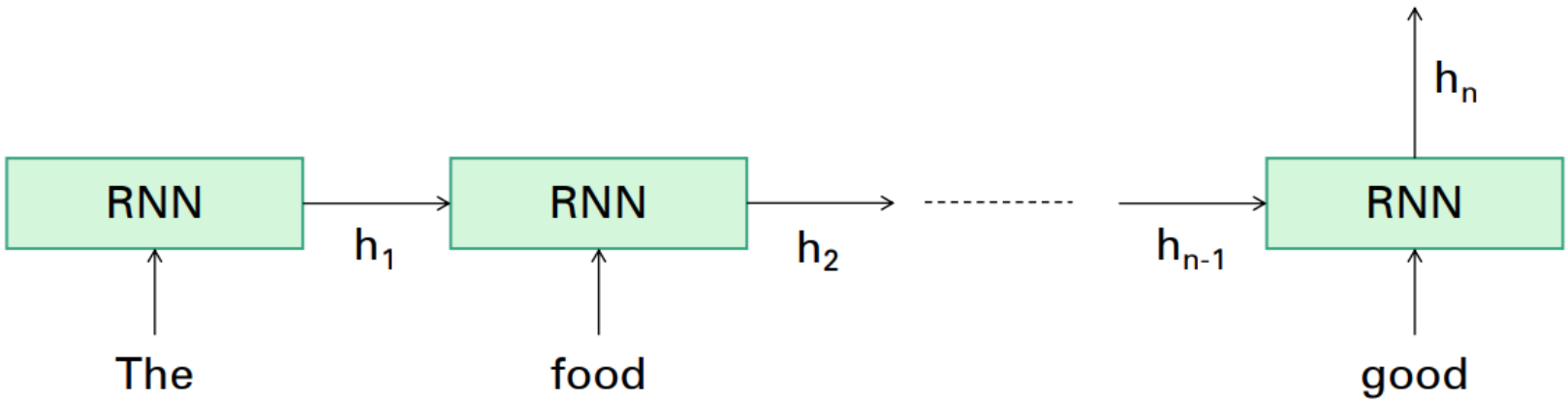
Sentiment Classification



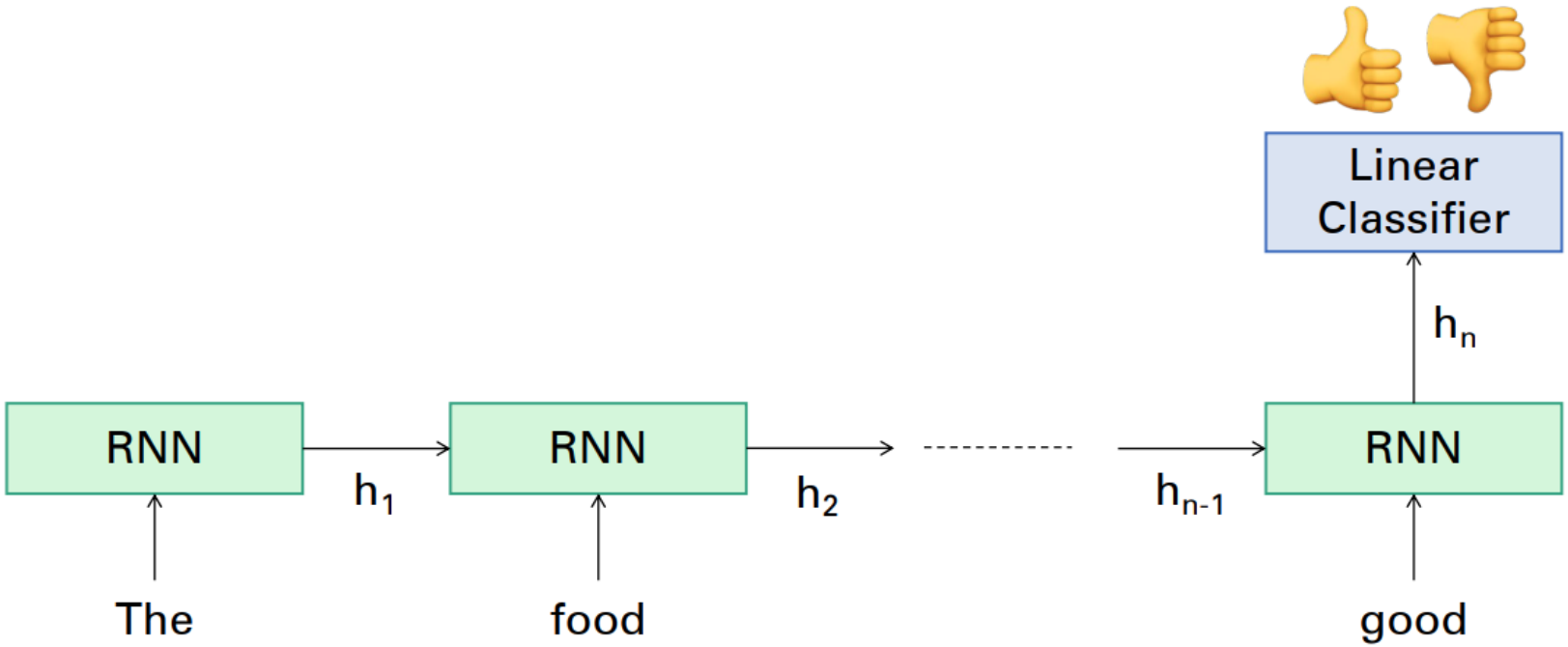
Sentiment Classification



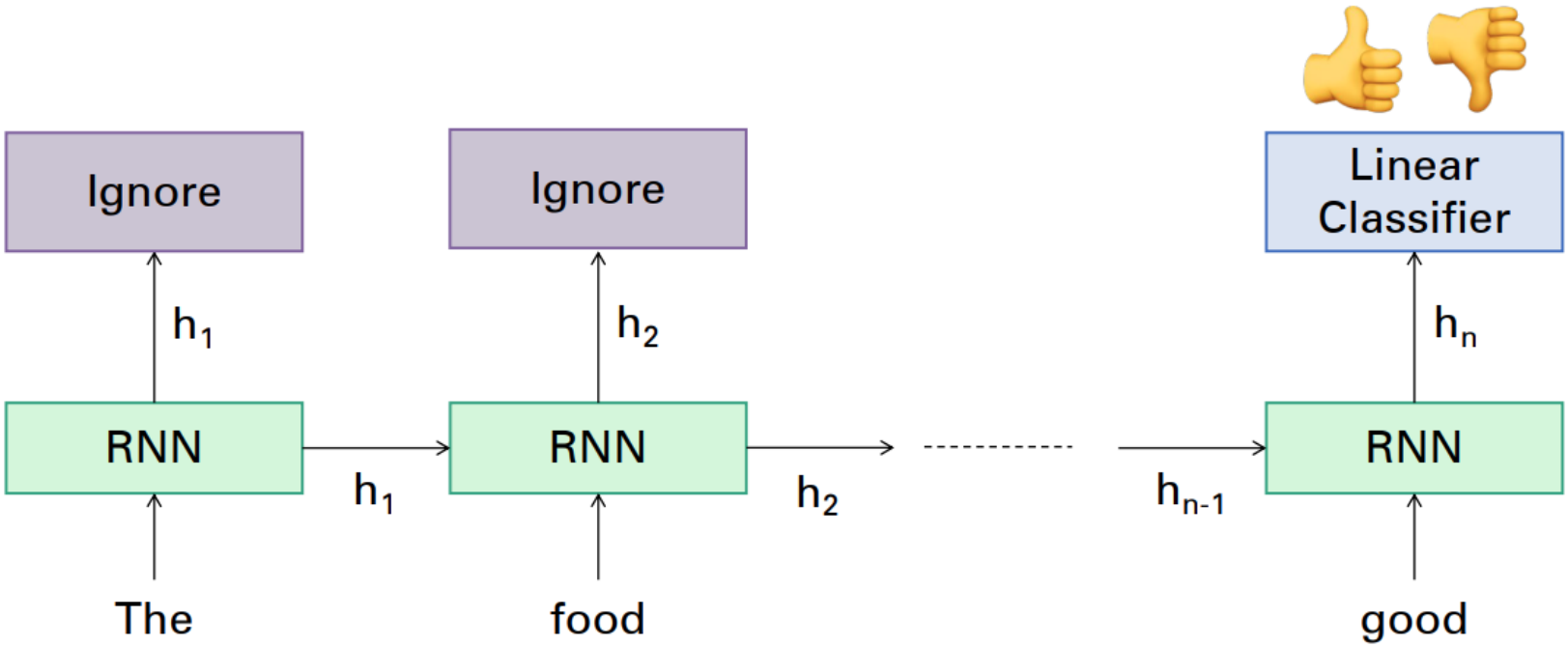
Sentiment Classification



Sentiment Classification



Sentiment Classification

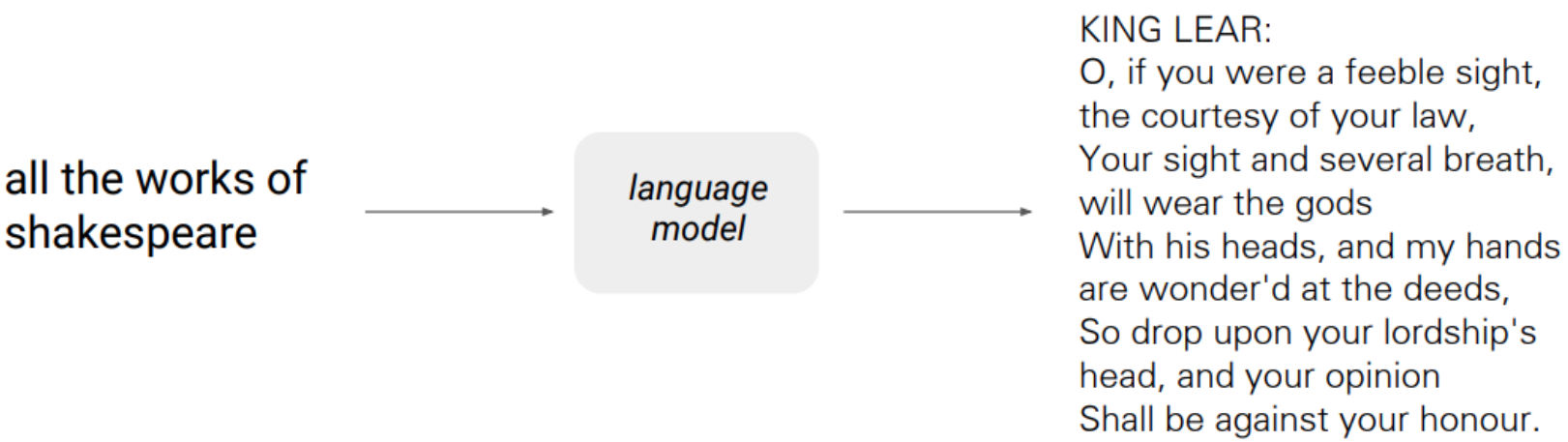


Applications or Examples

Language Modeling

Language Modeling

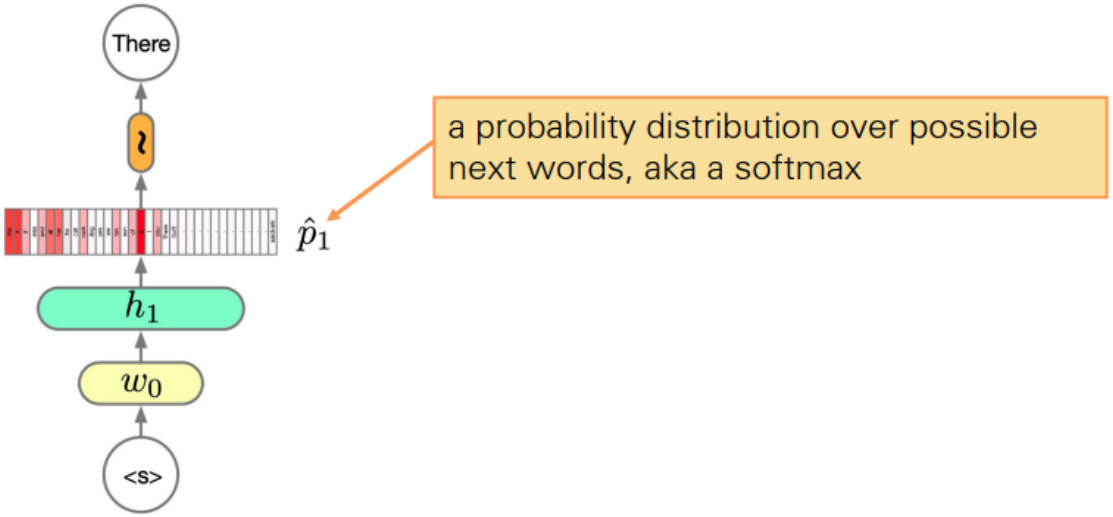
- Language models aim to represent the history of observed text $(w_1, ..., w_{t-1})$ succinctly in order to predict the next word (w_t):



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

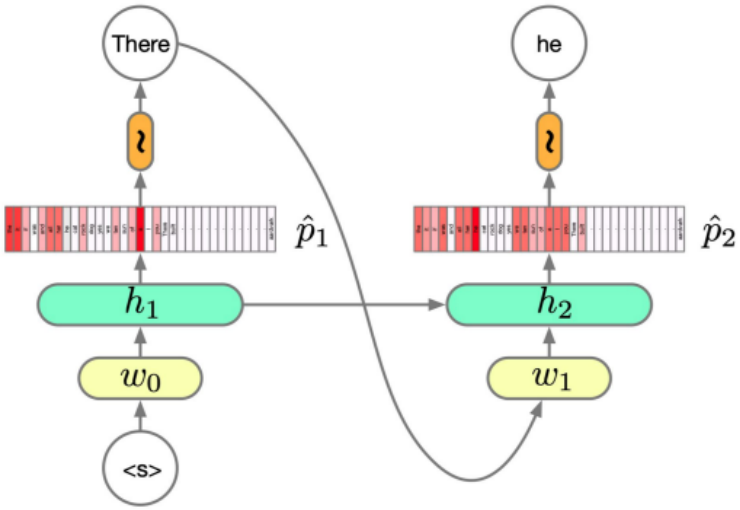
RNN Language Models

$$h_n = g(V\,[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = W h_n + b$$



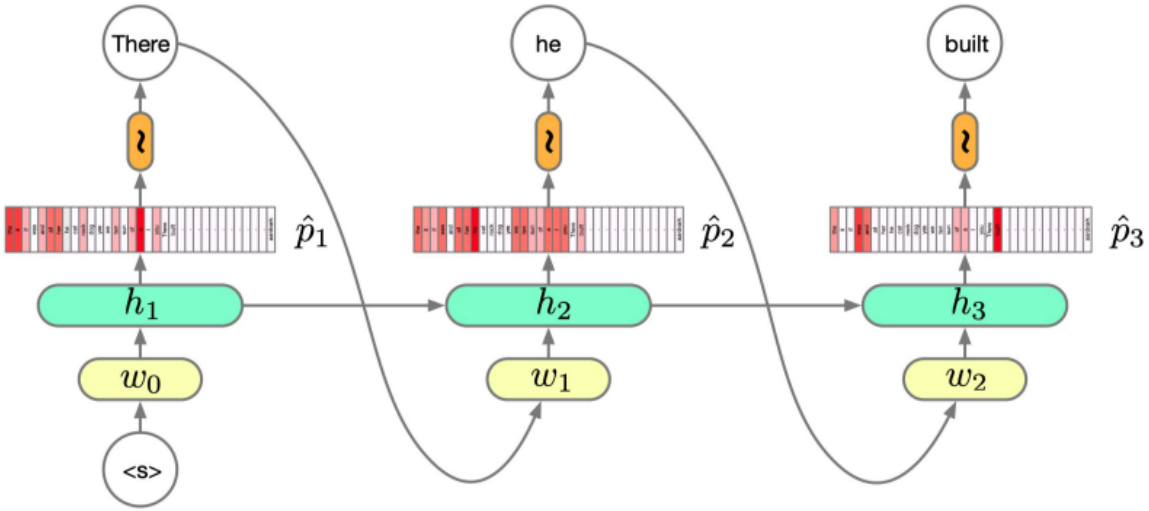
RNN Language Models

$$h_n = g(V\,[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = W h_n + b$$



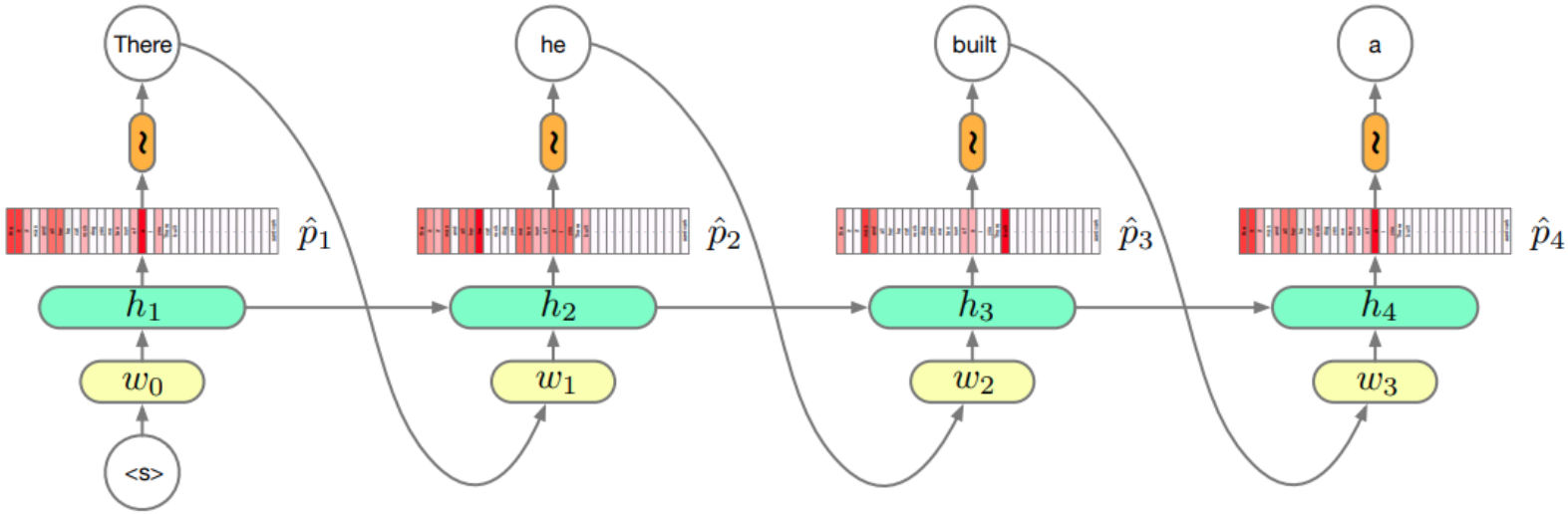
RNN Language Models

$$h_n = g(V\,[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = W h_n + b$$



RNN Language Models

$$h_n = g(V\,[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = W h_n + b$$



Applications or Examples

Image Captioning

Image Captioning

$$h_n = g(V\,[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = W h_n + b$$

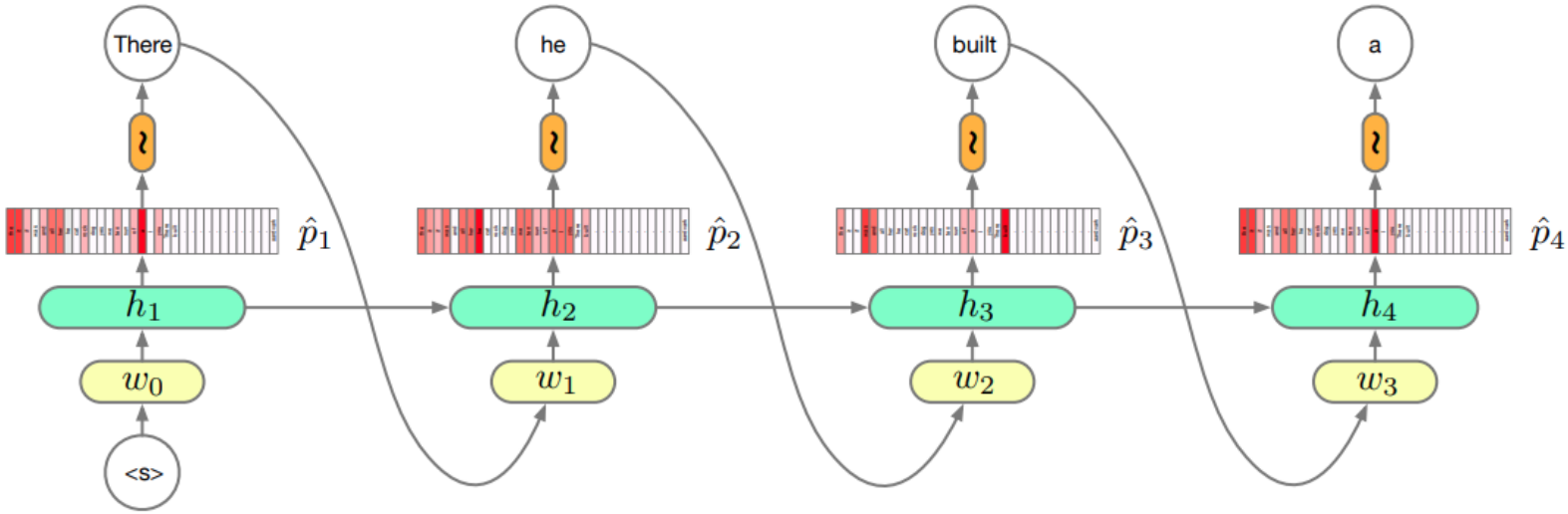
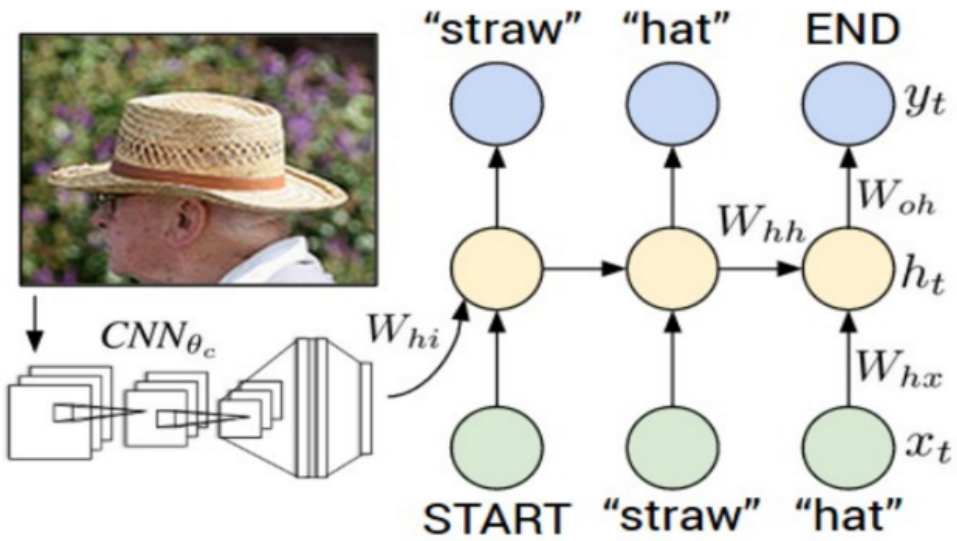


Image Captioning



Explain Images with Multimodal Recurrent Neural Networks [Mao et al.]
Deep Visual-Semantic Alignments for Generating Image Descriptions [Karpathy and Fei-Fei]
Show and Tell: A Neural Image Caption Generator [Vinyals et al.]
Long-term Recurrent Convolutional Networks for Visual Recognition and Description [Donahue et al.]
Learning a Recurrent Visual Representation for Image Caption Generation [Chen and Zitnick]

Image Captioning

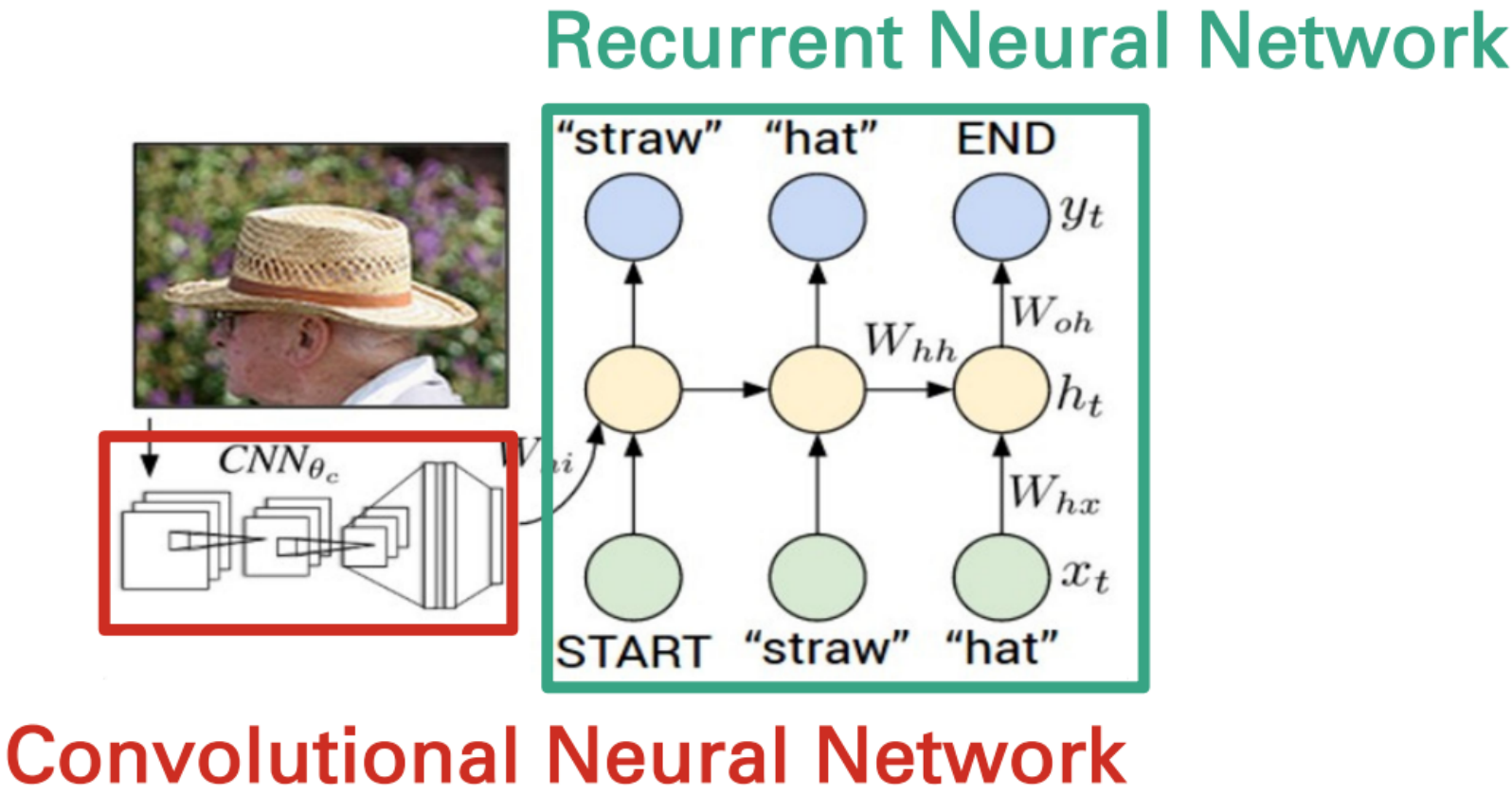


Image Captioning

test image

Image Captioning

Image Captioning

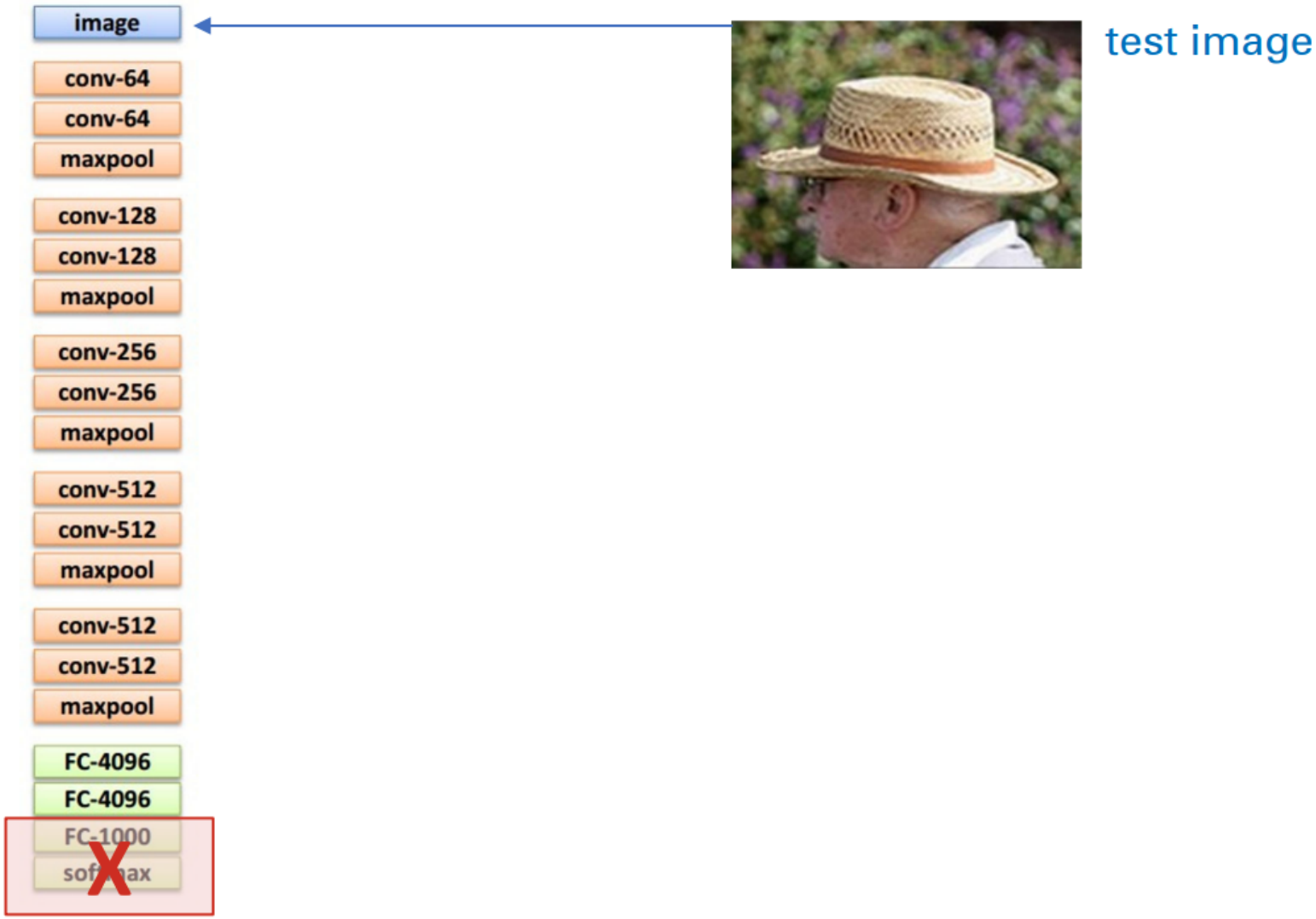


Image Captioning

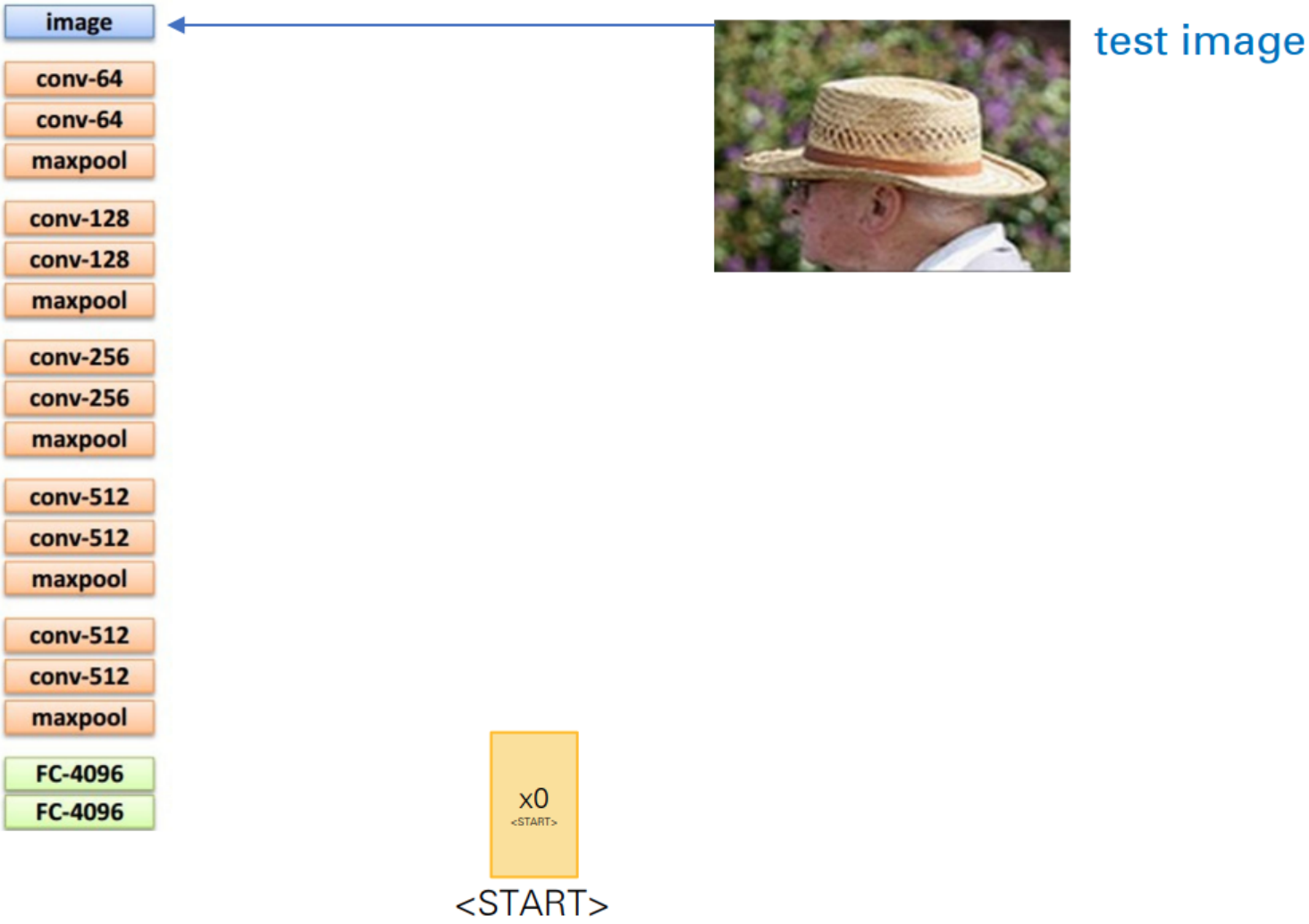
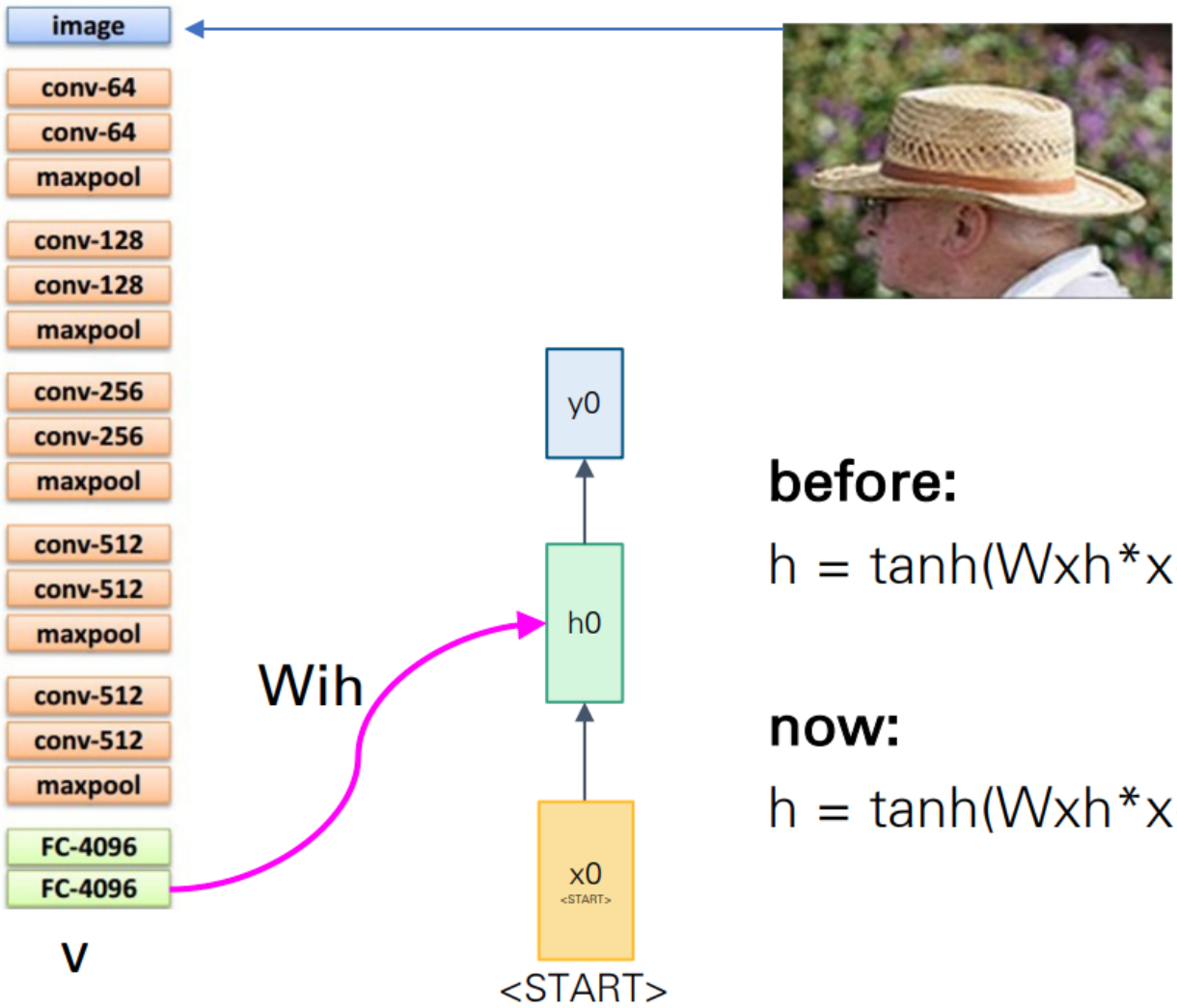


Image Captioning



before:

h = tanh(Wxh*x+Whh*h)

now:

h = tanh(Wxh*x+Whh*h+**Wih*v**)

Image Captioning

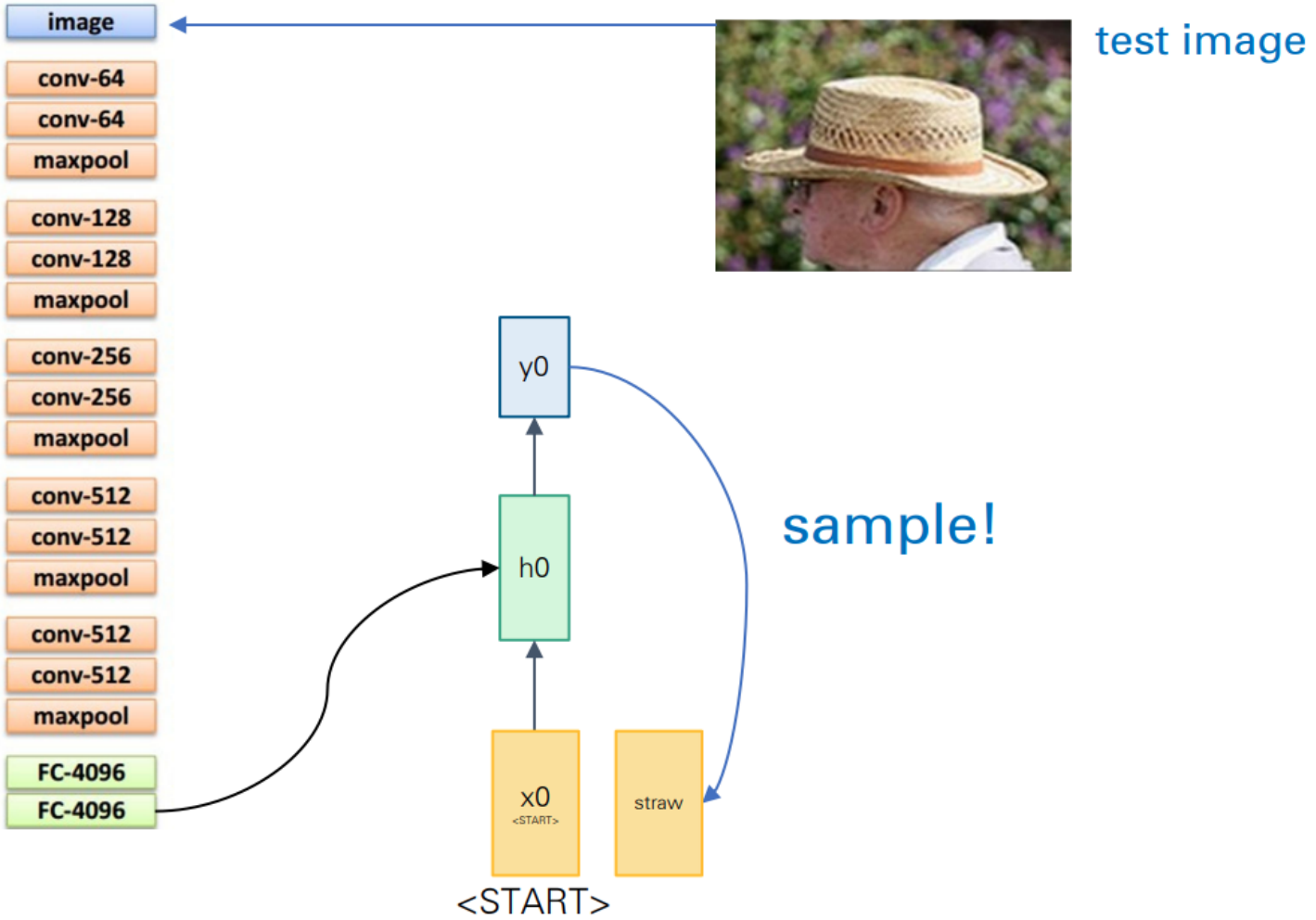


Image Captioning

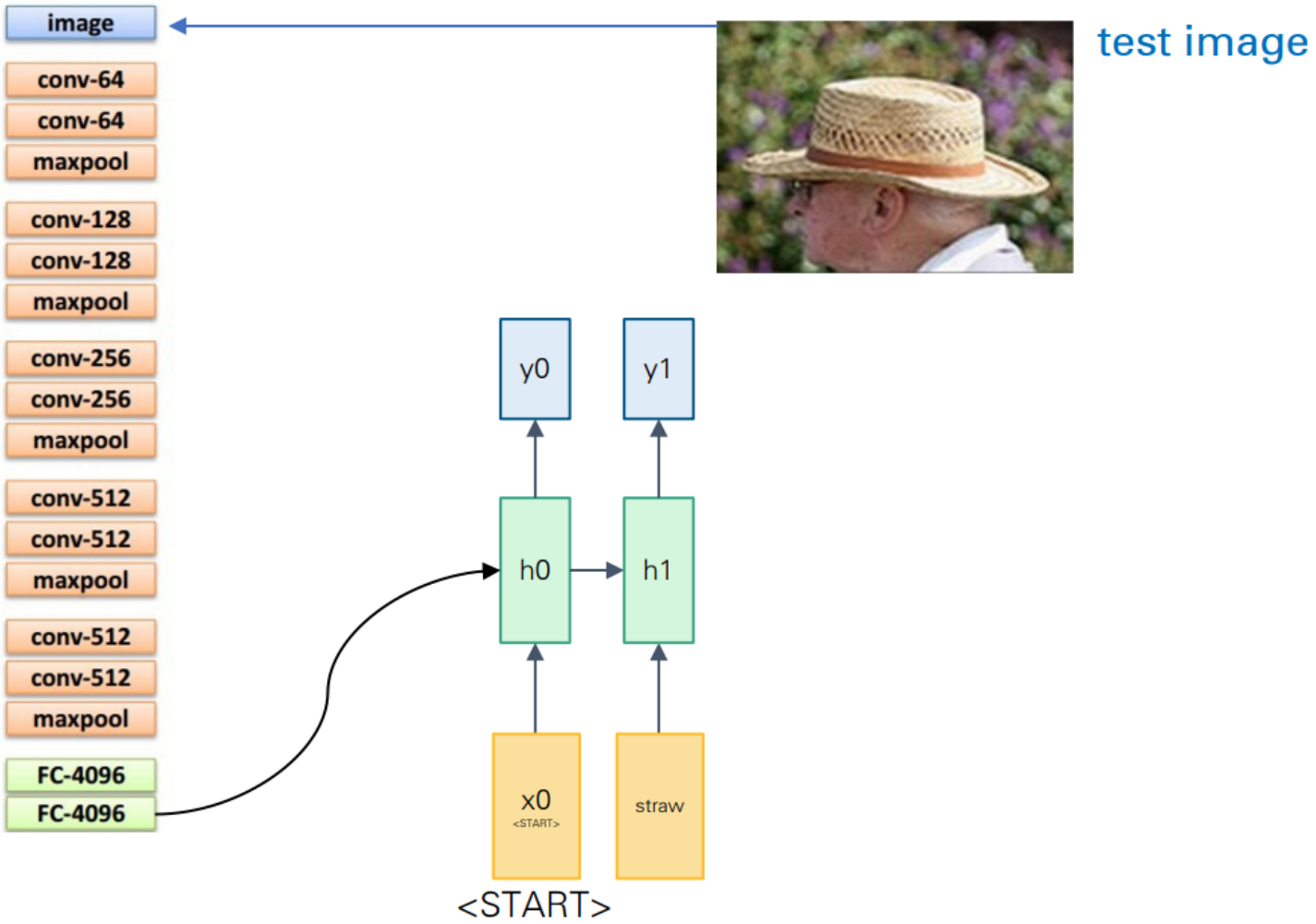


Image Captioning

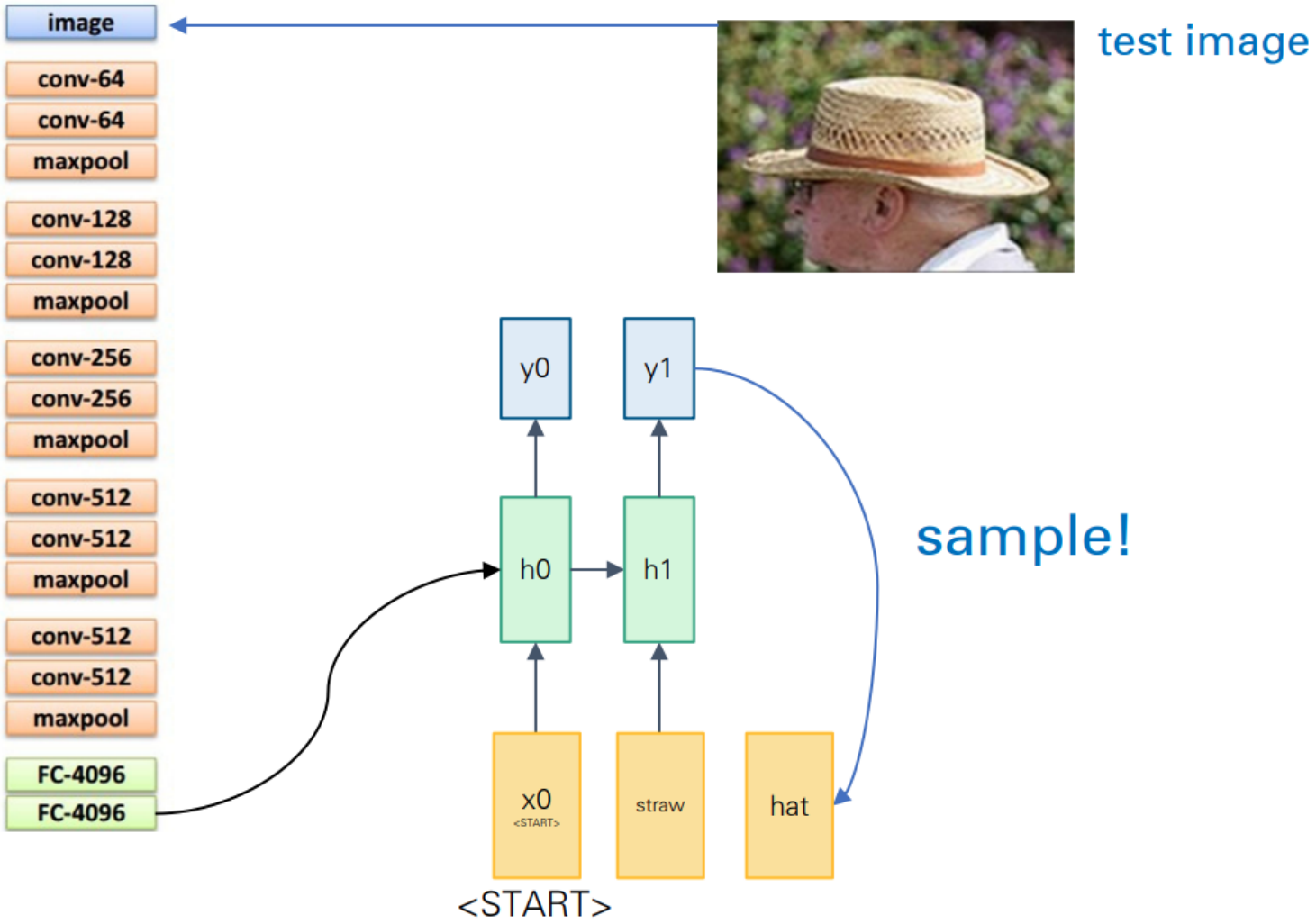


Image Captioning

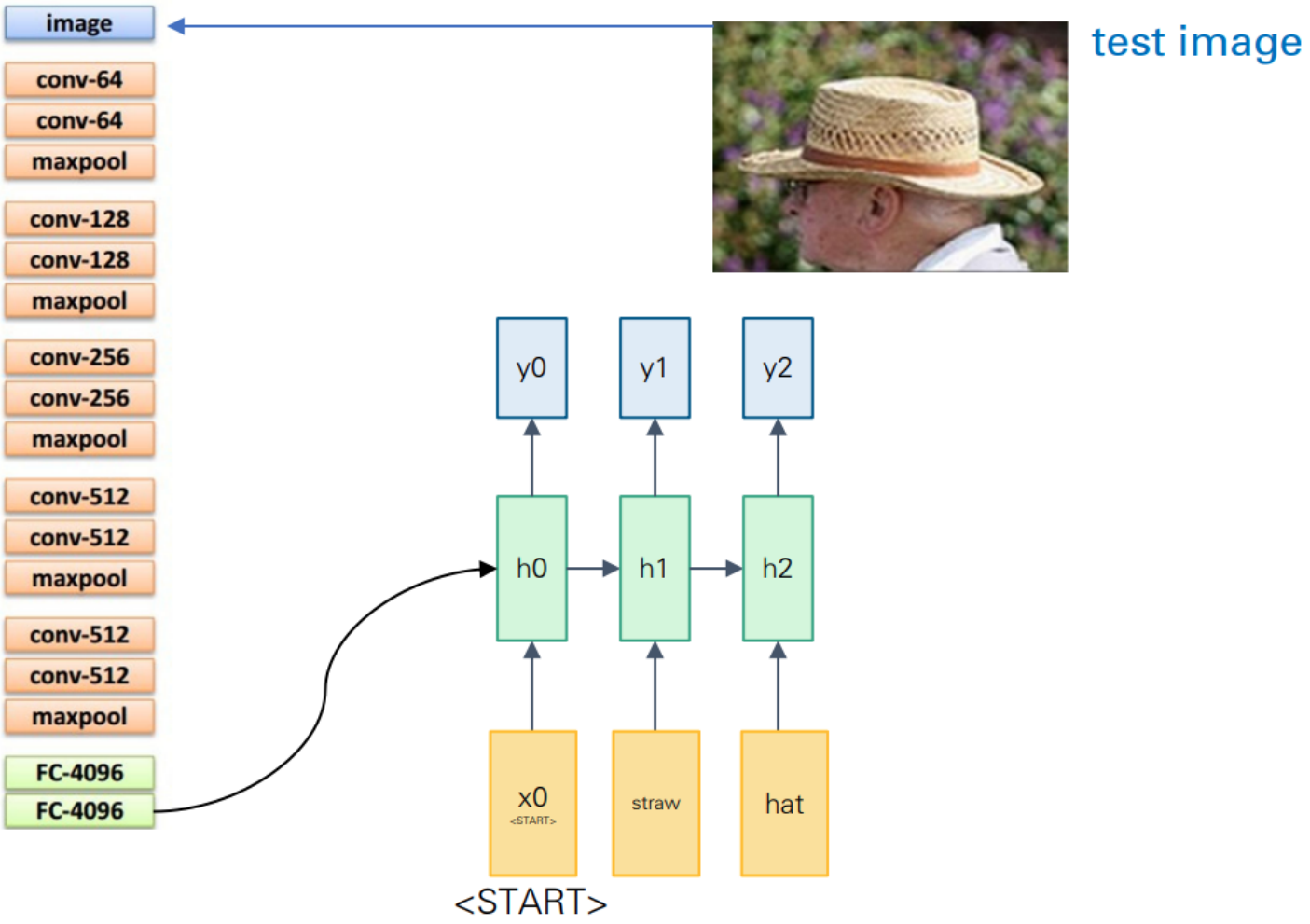
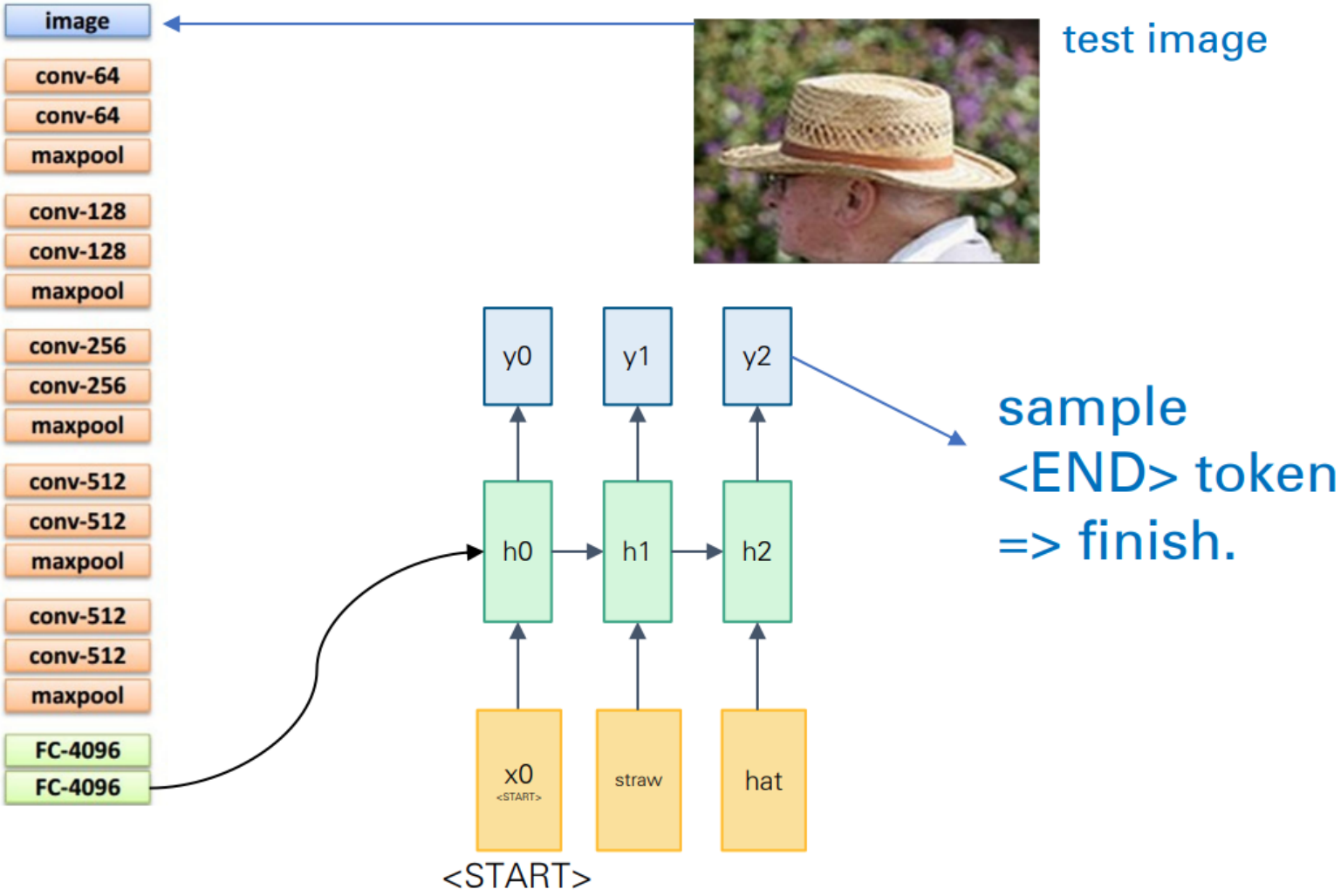
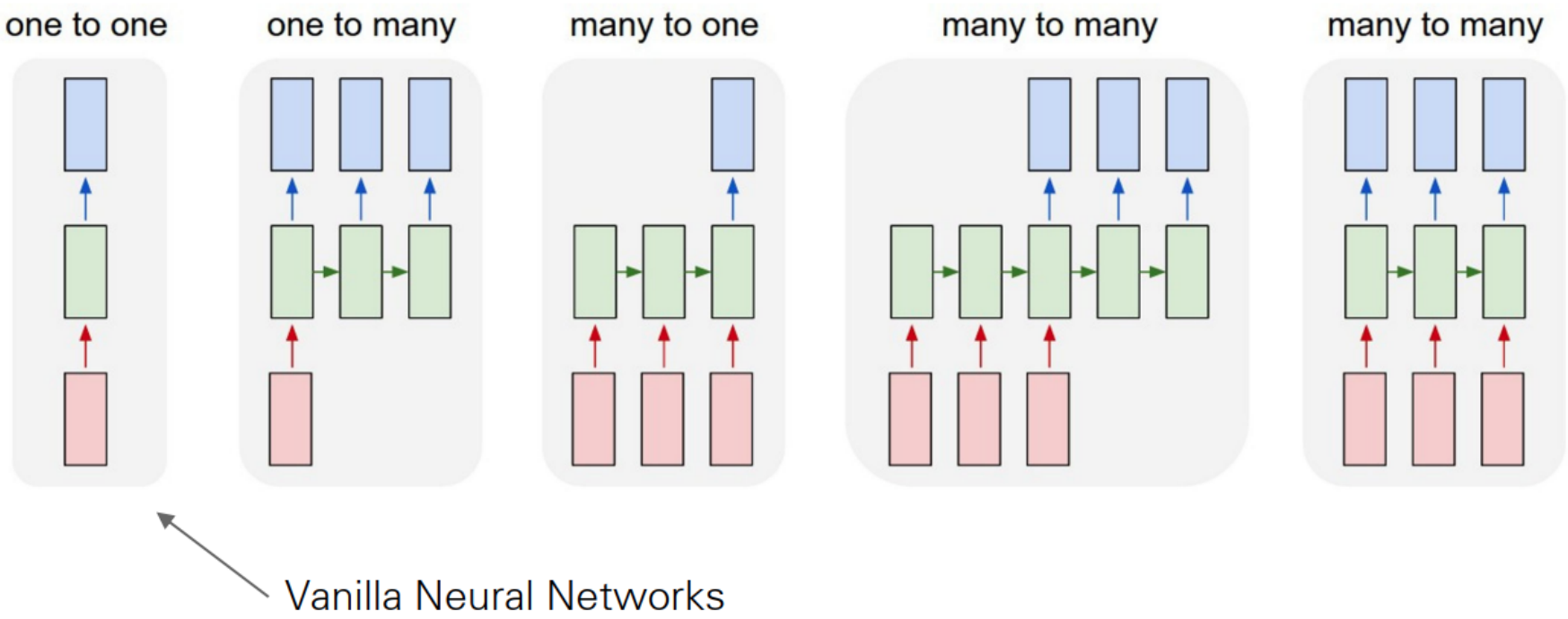


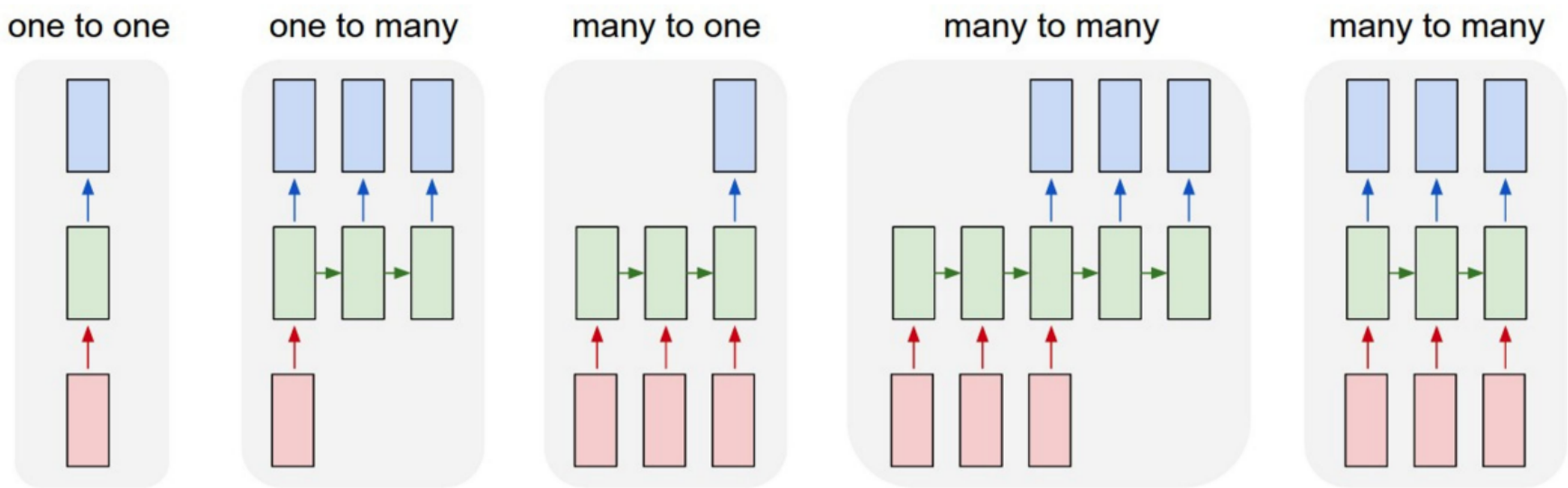
Image Captioning



Recurrent Networks offer a lot of flexibility

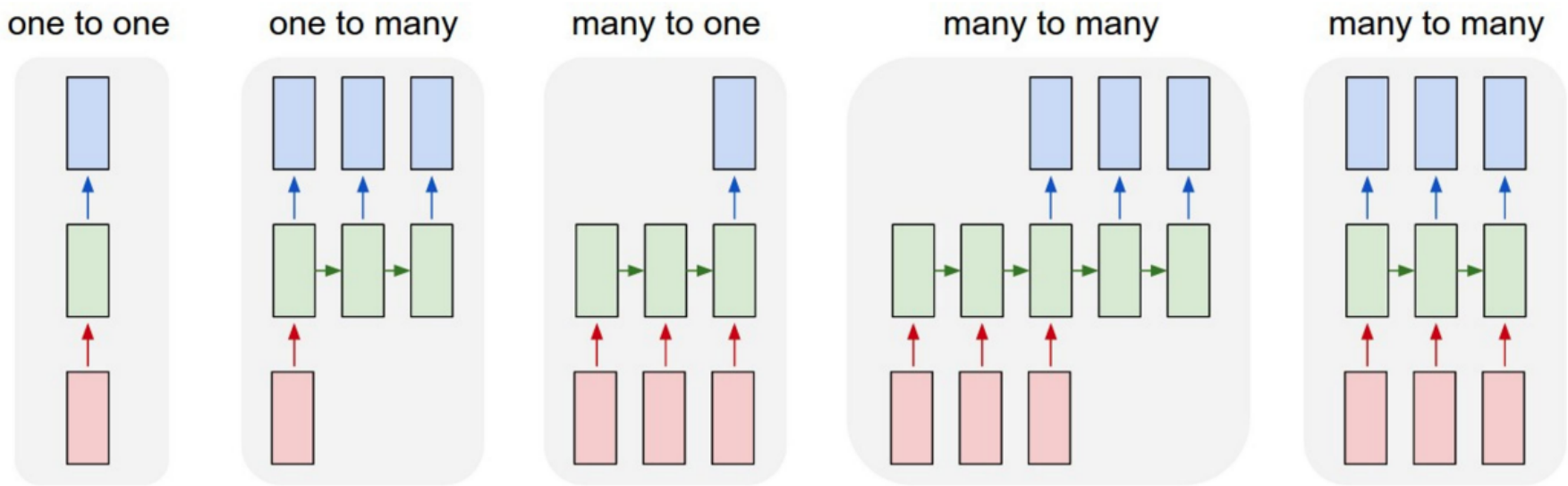


Recurrent Networks offer a lot of flexibility



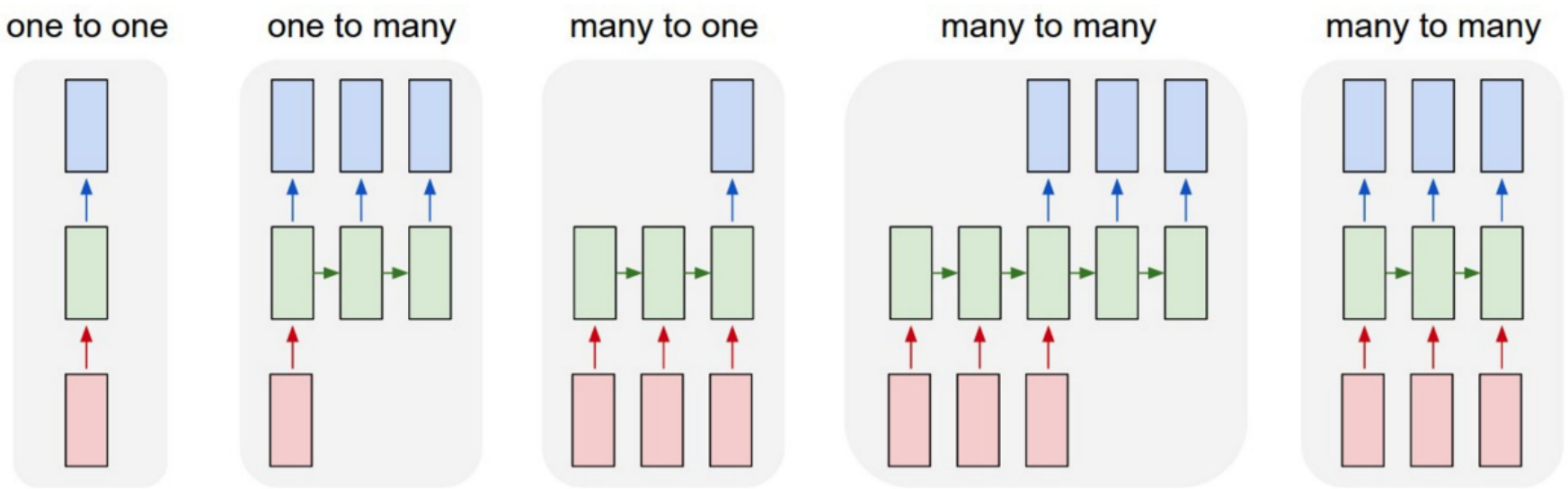
e.g. **Image Captioning**
image -> sequence of words

Recurrent Networks offer a lot of flexibility

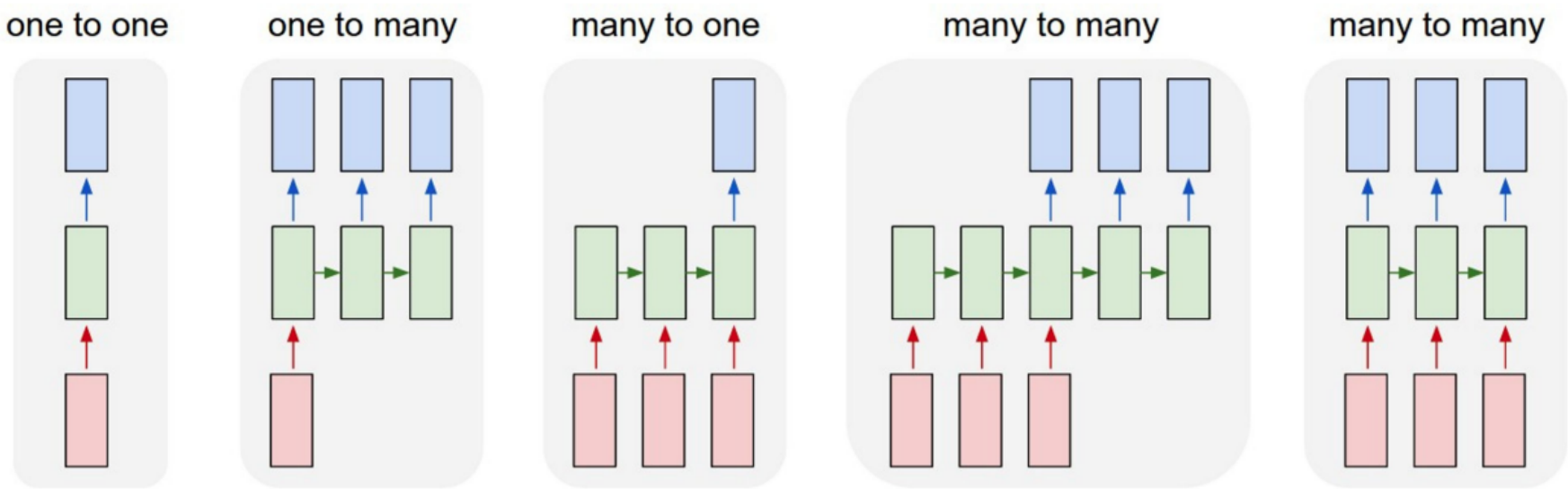


e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Networks offer a lot of flexibility

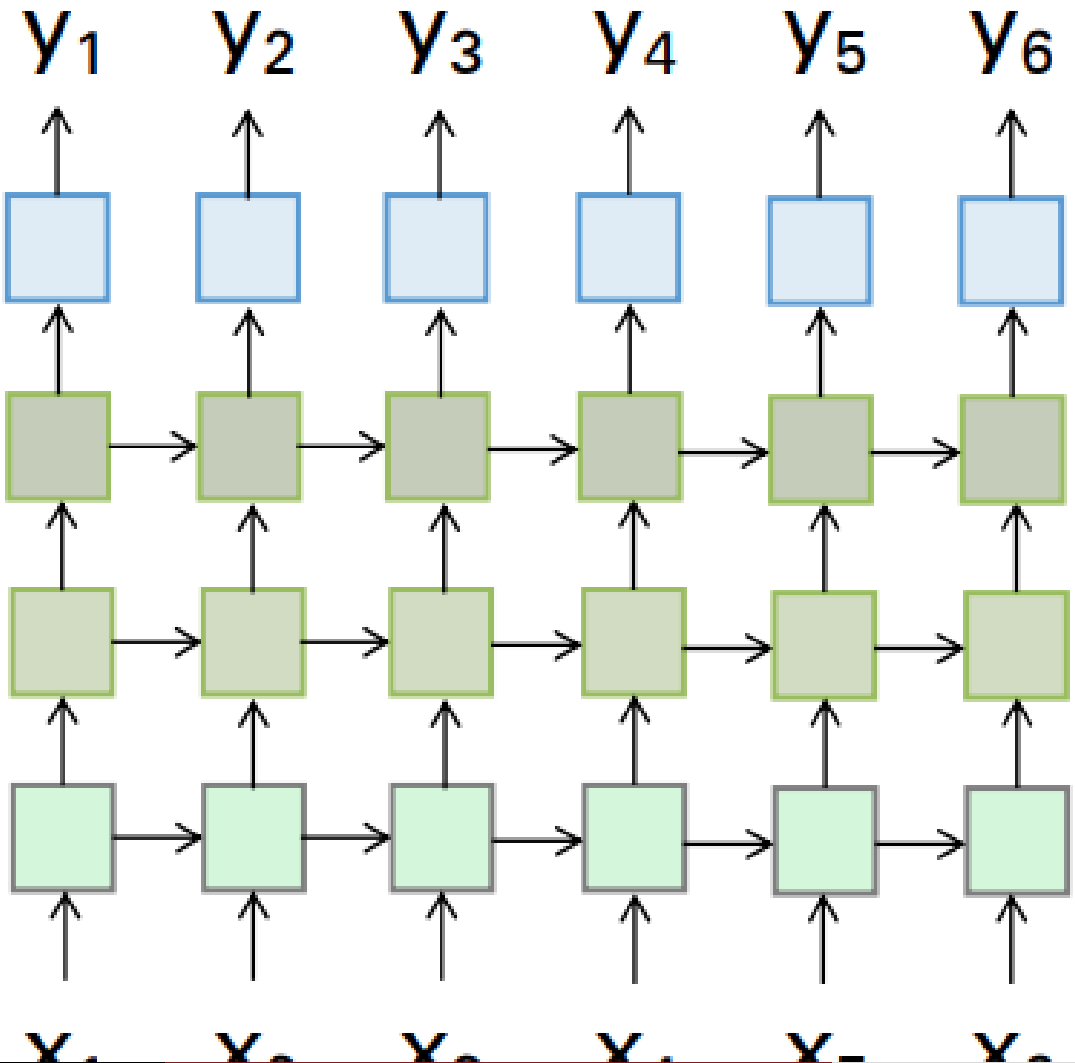


Recurrent Networks offer a lot of flexibility



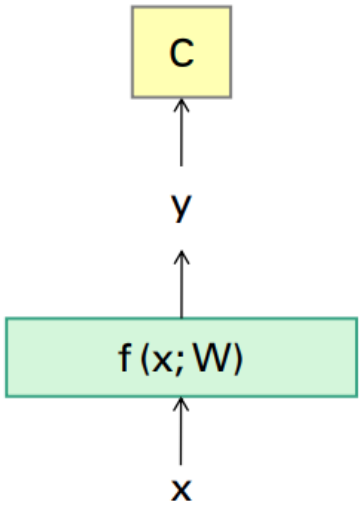
Multi-layer RNNs

- We can of course design RNNs with multiple hidden layers



How to Train Recurrent Neural Networks

BackPropagation Refresher



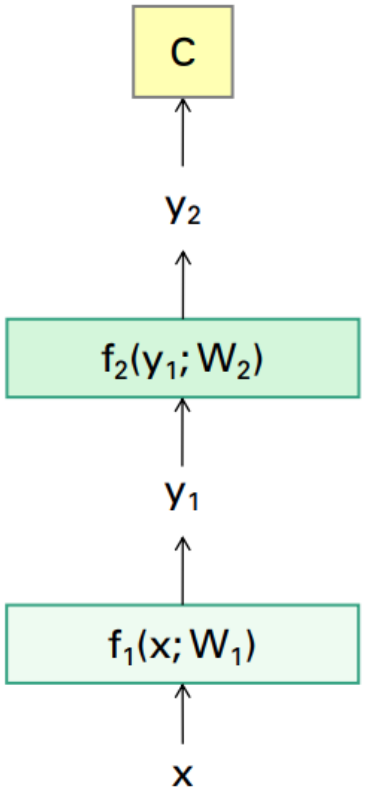
$$y = f(x; W)$$
$$C = \text{Loss}(y, y_{GT})$$

SGD Update

$$W \leftarrow W - \eta \frac{\partial C}{\partial W}$$

$$\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right)$$

Multiple Layers



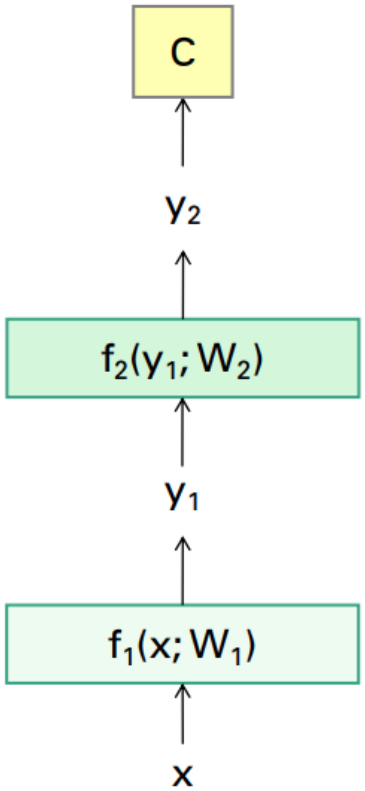
$$y_1 = f_1(x; W_1)$$
$$y_2 = f_2(y_1; W_2)$$
$$C = \text{Loss}(y_2, y_{GT})$$

SGD Update

$$W_2 \leftarrow W_2 - \eta \frac{\partial C}{\partial W_2}$$

$$W_1 \leftarrow W_1 - \eta \frac{\partial C}{\partial W_1}$$

Chain Rule for Gradient Computation



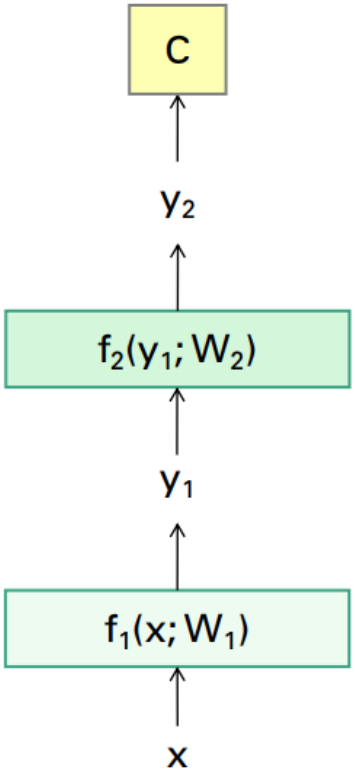
$$y_1 = f_1(x; W_1)$$
$$y_2 = f_2(y_1; W_2)$$
$$C = \text{Loss}(y_2, y_{GT})$$

SGD Update

$$W_2 \leftarrow W_2 - \eta \frac{\partial C}{\partial W_2}$$

$$W_1 \leftarrow W_1 - \eta \frac{\partial C}{\partial W_1}$$

Chain Rule for Gradient Computation



$$y_1 = f_1(x; W_1)$$
$$y_2 = f_2(y_1; W_2)$$
$$C = \text{Loss}(y_2, y_{GT})$$

Find $\frac{\partial C}{\partial W_1}, \frac{\partial C}{\partial W_2}$

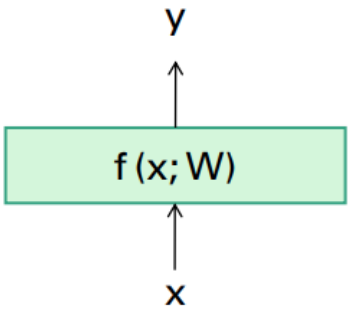
$$\frac{\partial C}{\partial W_2} = \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial W_2}\right)$$

$$\frac{\partial C}{\partial W_1} = \left(\frac{\partial C}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

$$= \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

Application of the Chain Rule

Chain Rule for Gradient Computation



Given: $\left(\frac{\partial C}{\partial y}\right)$

We are interested in computing: $\left(\frac{\partial C}{\partial W}\right), \left(\frac{\partial C}{\partial x}\right)$

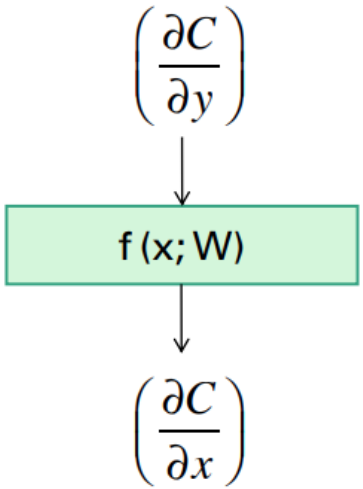
Intrinsic to the layer are:

$\left(\frac{\partial y}{\partial W}\right)$ – How does output change due to params

$\left(\frac{\partial y}{\partial x}\right)$ – How does output change due to inputs

$$\left(\frac{\partial C}{\partial W}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right) \quad \left(\frac{\partial C}{\partial x}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial x}\right)$$

Chain Rule for Gradient Computation



Given: $\left(\frac{\partial C}{\partial y}\right)$

We are interested in computing: $\left(\frac{\partial C}{\partial W}\right), \left(\frac{\partial C}{\partial x}\right)$

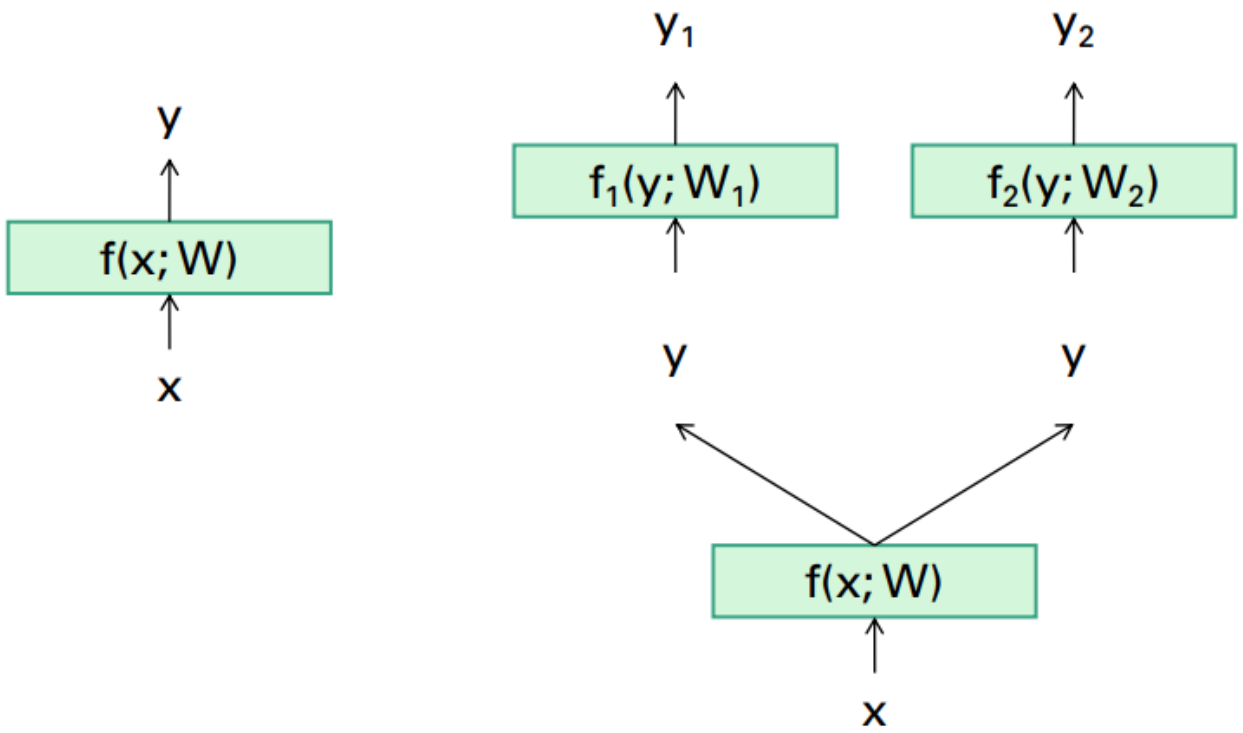
Intrinsic to the layer are:

$\left(\frac{\partial y}{\partial W}\right)$ – How does output change due to params

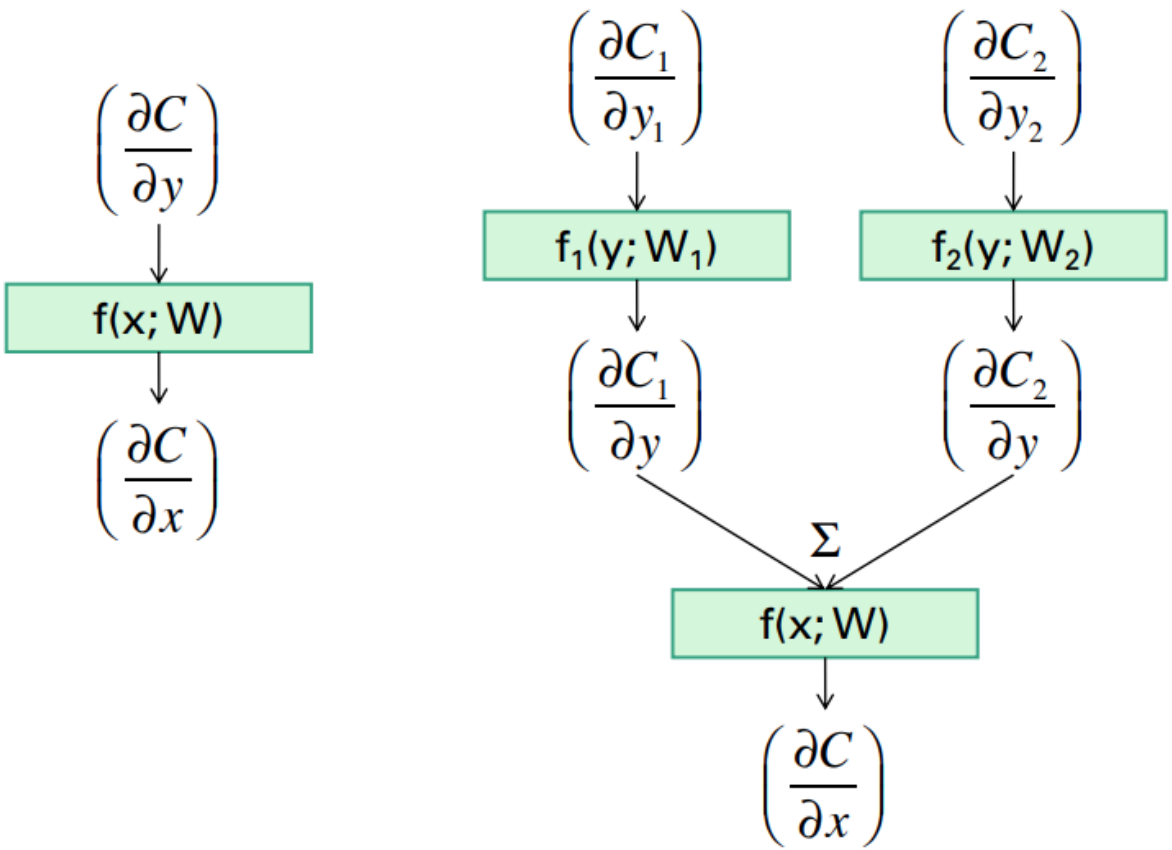
$\left(\frac{\partial y}{\partial x}\right)$ – How does output change due to inputs

$$\left(\frac{\partial C}{\partial W}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right) \quad \left(\frac{\partial C}{\partial x}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial x}\right)$$

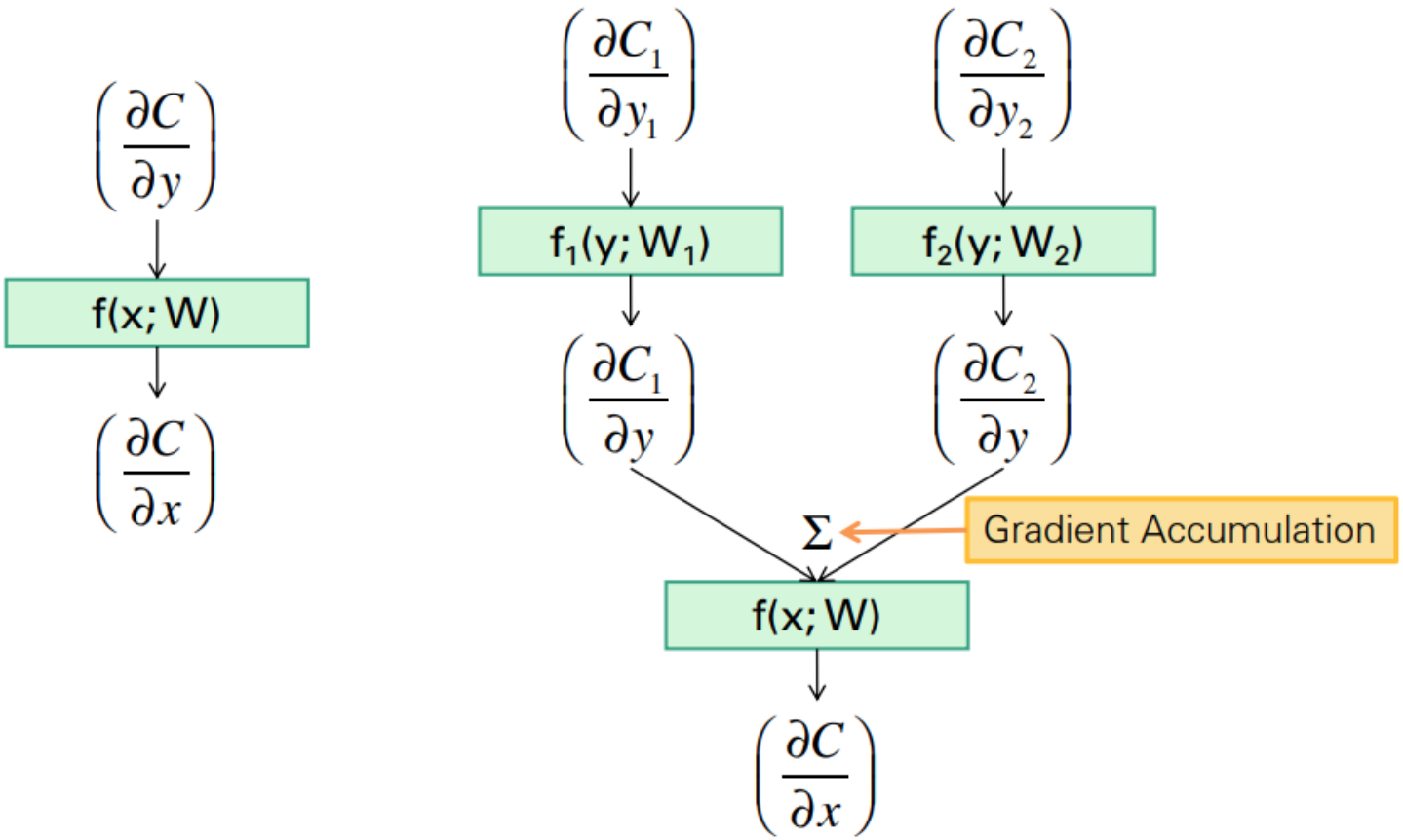
Extension to Computational Graphs



Extension to Computational Graphs

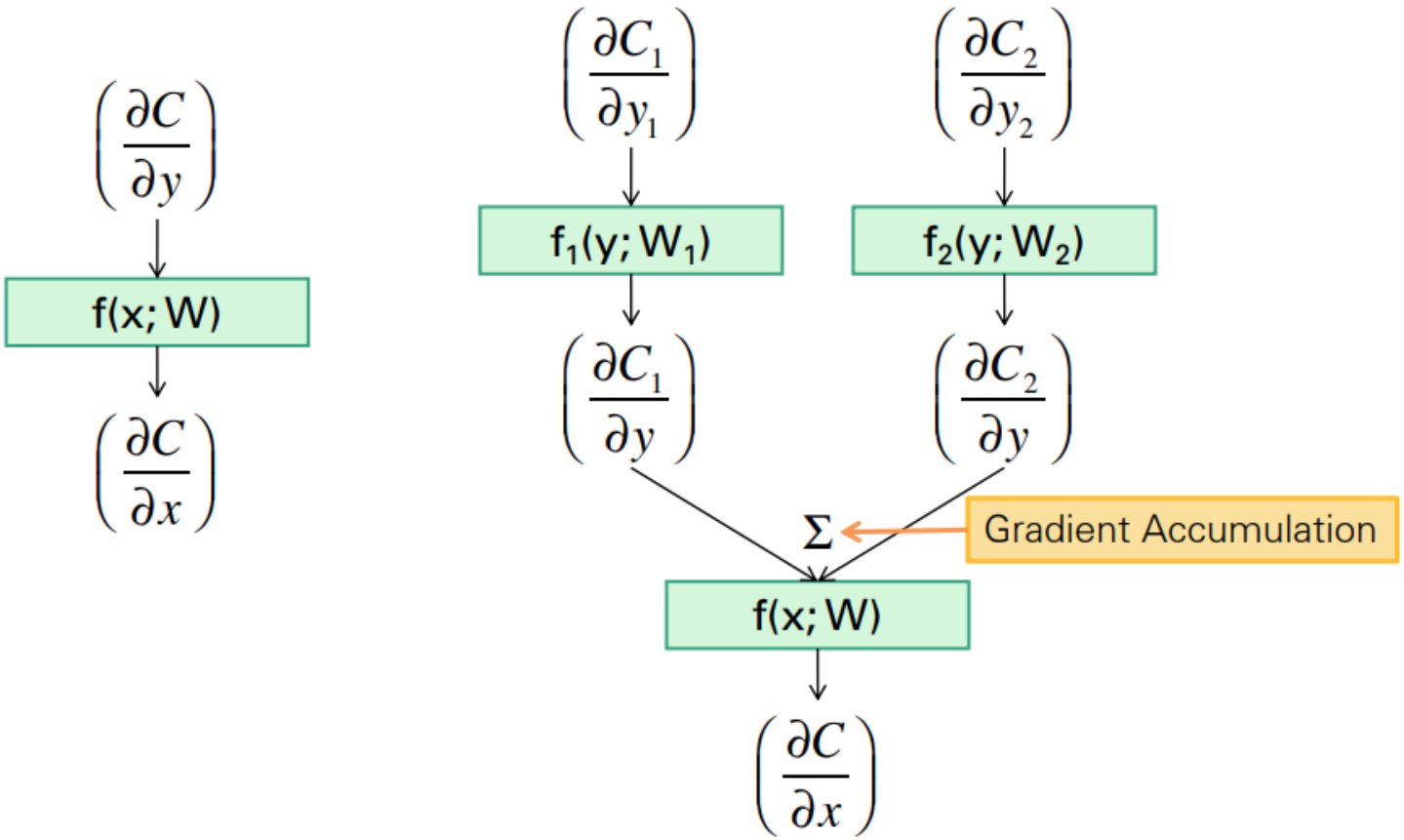


Extension to Computational Graphs



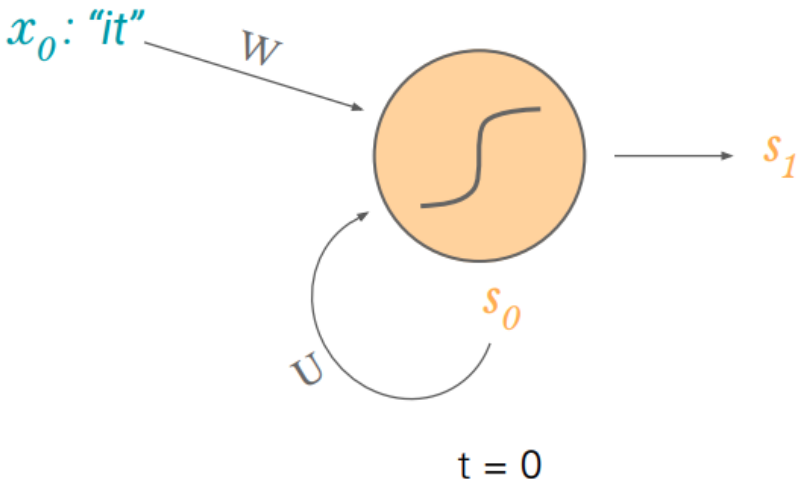
Sample RNN

- RNNs remember their previous state:



Sample RNN

- RNNs remember their previous state:



x_0 : vector representing first word
s_0 : cell state at $t = 0$ (some initialization)
s_1 : cell state at $t = 1$

$$s_1 = tanh(Wx_0 + Us_0)$$

W, U : weight matrices

RNN, LSTMs and GRU

Next Class..